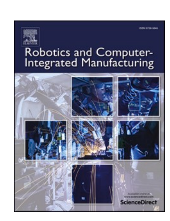
Robotics and Computer–Integrated Manufacturing 91 (2025) 102851

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim





Smooth joint motion planning for redundant fiber placement manipulator based on improved RRT*

Qian Yang [a,b], Weiwei Qu [a,b,*], Yanzhe Wang [a,b], Xiaowen Song [a,b], Yingjie Guo [a,b], Yinglin Ke [a,b]

[a] State Key Laboratory of Fluid Power and Mechatronic Systems, School of Mechanical Engineering, Zhejiang University, Hangzhou, 310027, China
[b] Key Laboratory of Advanced Manufacturing Technology of Zhejiang Province, School of Mechanical Engineering, Zhejiang University, Hangzhou, 310027, China

ARTICLE INFO

Keywords:
Kinematic redundancy
Smooth joint motion
Redundancy optimization
Automated fiber placement

ABSTRACT

In automated fiber placement (AFP), addressing the continuous motion planning challenge of redundant layup manipulators in complex environments, this paper proposes an offline redundancy optimization algorithm based on improved RRT* (Rapidly-exploring Random Trees). This algorithm maximizes the utilization of kinematic redundancy to derive smooth joint trajectories devoid of collisions and singularities. Firstly, the algorithm entails constructing a search map by eliminating joint configurations that violate constraints, and subsequently planning and optimizing the joint path by minimizing a multi-objective cost under the map constraint. Furthermore, several strategies are introduced to enhance RRT* for redundancy optimization. These strategies include a piecewise Gaussian sampling strategy (PGSS) to guide efficient tree growth within complex channels and enable joint sampling constrained by task coordinates. Additionally, the improved Steering and Local Optimization method are proposed to plan joint motion while considering intermediate task sequences. The effectiveness of the proposed algorithm is demonstrated in handling complex motion planning scenarios, such as layup involving complex path curves or dense obstacles. Experimental results validate the algorithm's capability to find feasible collision-free and singularity-free paths in relevant scenarios, provided such paths exist. Moreover, trajectory smoothness is optimized with increasing iterations.

1. Introduction

Carbon fiber composite components have superior properties such as specific stiffness, strength, and corrosion resistance [1]. It is clear that their use in aerospace applications is widespread, and the automation of the manufacturing process is beneficial. The two cornerstone automated processes are automated tape laying (ATL) and automated fiber placement (AFP). The latter is the focus here. The ATL process replaces the manual layup of unidirectional tapes (75–300 mm wide [2]) but can be done at higher speeds, on larger parts, and with better control over slightly curved surfaces [3,4]. AFP offers a better adaptation to complex surfaces. The AFP process comprises a machine/robot system and an attached fiber placement head. The AFP head can lay multiple tows onto the mold surface. Adhesion between the tows and the substrate is ensured using appropriate process conditions (e.g., heating, compaction, and tensioning systems). Typically, a group of tows forms a layup path; these paths are combined to create a layer; multiple layers form a laminate.

The fiber placement task necessitates 6 degrees of freedom (DOFs) for the layup device to determine the head's position and orientation, including compaction and laying direction (as depicted in Fig. 1). If the layup manipulator possesses more than 6 DOFs, it results in kinematic redundancy. This redundancy allows for various joint configurations while maintaining the same layup position and orientation of the fiber placement head. Leveraging this redundancy can optimize the manipulator's joint trajectory [5-8]. Given the layup path, kinematic model, manipulator geometry model, and machining environment, redundancy optimization produces a sequence of joint configurations that track the desired path and adhere to specific constraints. In the context of AFP, redundancy optimization must also consider the smoothness of joint motion.

Smooth joint motion enhances layup efficiency and speed stability, effectively mitigating machine vibrations that may compromise machine lifespan and layup accuracy. Velocity and acceleration significantly affect joint motion smoothness. Specifically, when the end feed rate of the AFP machine is determined, motion characterized by lower joint velocities and accelerations allows for greater speed-up margins and reduced vibration. Consequently, redundancy optimization should

* Corresponding author.
E-mail address: qwwwwl@zju.edu.cn (W. Qu).

https://doi.org/10.1016/j.rcim.2024.102851
Received 21 December 2022; Received in revised form 23 July 2024; Accepted 2 August 2024
Available online 1 September 2024
0736-5845/© 2024 Elsevier Ltd. All rights reserved, including those for text and data mining, AI training, and similar technologies.

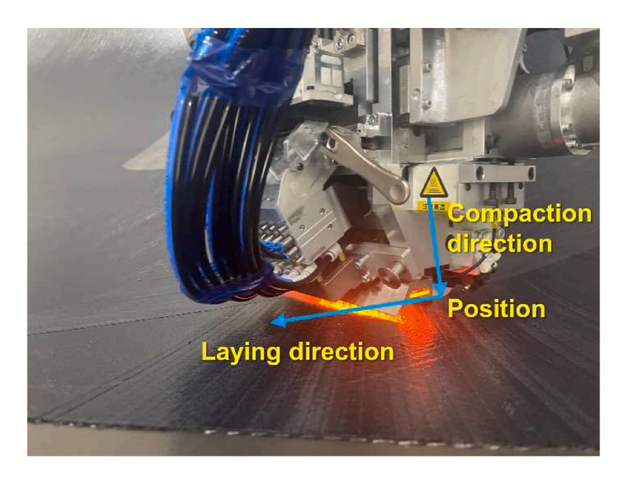


Fig. 1. Fiber placement task.

prioritize motion performance in terms of velocity and acceleration.

Meanwhile, the joint trajectory is expected to follow some constraints, such as collision and singularity avoidance. One challenge for avoiding collisions between manipulator links and the mold is the computational cost of collision detection. Researchers commonly use the bounding volume technology to reduce computational effort [9]. The core idea of bounding volume technology is to approximately describe an object with an envelope whose volume completely contains the object and is slightly larger than the object. Typical bounding volume techniques mainly include the bounding sphere, aligned axis bounding box (AABB), oriented bounding box (OBB), and other kinds of bounding volumes [10]. Singularity avoidance is another essential constraint for the manipulator motion. The singularity is located at a specific position in the workspace of the manipulator, and the manipulator will lose one degree of freedom at the singular pose. Near the singularity point, the joint changes dramatically [11].

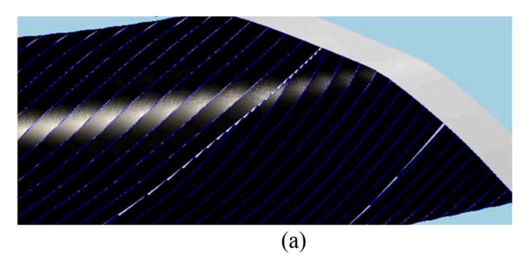
In redundancy optimization, manipulator motion typically involves manual teaching, online programming, or offline planning. Manual teaching programming entails optimizing robot trajectories based on specific task requirements through action demonstrations and regression coding. Online programming, on the other hand, typically utilizes integrated cameras and laser sensors to gather information on the manipulator's current position and surrounding environment, generating real-time commands to ensure safe manipulation to the designated position. For AFP applications, the manipulator end must traverse a predetermined path established by the designer under various process constraints, as illustrated in Fig. 2. This path dictates layup quality and cannot be arbitrarily adjusted. Furthermore, since layup manufacturing entails continuous tow placement, maintaining motion continuity of the manipulator is crucial to prevent issues such as twisted tows and wrinkles. Consequently, online programming is not suitable for layup motion planning involving fixed-end paths, high accuracy requirements, and continuity needs. Moreover, manual teaching is impractical due to its high time cost, especially given the multiple task paths and diverse path geometries involved. Hence, offline automatic motion planning (OLAMP) emerges as the most suitable planning mode under these circumstances. For instance, techniques like Dijkstra's shortest path [12-14] transform the continuous problem into a discrete one and optimize automatic joint configurations. Although OLAMP algorithms effectively address existing challenges to some extent, complexities arise when dealing with complex molds or environments. Multiple constraints result in widely scattered feasible joint solution distributions, leading to difficulties in modeling and computational inefficiencies. Therefore, there is a pressing need to develop more efficient redundant manipulator motion planning methods for complex surfaces.

In the field of manipulator motion planning, Sample-based algorithms (e.g., RRT [15], RRT* [16], PRM* [17], FMT* [18], etc.) have high efficiency. These algorithms sample in a high-dimensional joint space with simultaneous collision detection and can quickly deal with motion planning problems with non-complete constraints. According to the literature research, the Sample-based algorithms plan collision-free joint motions of a robot from a given starting point to the endpoint. These methods cannot be directly applied to layup motion planning due to the requirement of end-position constraints for the motion between the starting and endpoints. Therefore, besides addressing joint continuity and collision detection, it's essential to account for end-position constraints to ensure the accurate layup of the prepreg tape for layup motion planning.

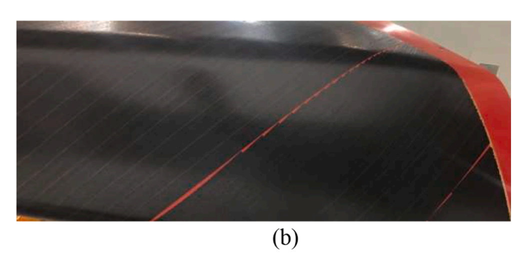
Therefore, this paper introduces an RRT*-based redundancy optimization algorithm designed to tackle the complex continuous motion planning challenges encountered in 7 DOFs redundant AFP systems. The proposed algorithm enhances the RRT* approach to generate smoother joint motion trajectories that accurately follow the continuous fiber placement path while avoiding collisions and singularities. Firstly, the algorithm shapes a search map by removing constraint-violating configurations, effectively characterizing the layup environment's complexity. Subsequently, the improved RRT* algorithm is employed for motion planning under these map constraints. To address the joint and task space constraints simultaneously, a piecewise Gaussian sampling strategy (PGSS) is introduced, guiding the tree expansion efficiently through complex channels. Moreover, recognizing that task coordinates impose constraints on joints, improvements are made in Steering and Local Optimization for joint motion planning constrained by intermediate tasks. Additionally, a path cost function is formulated to minimize joint velocity and acceleration. Overall, the proposed algorithm offers an effective solution to continuous layup motion planning challenges, particularly in complex environments where system redundancy is a consideration.

1.1. Related work

As composite component manufacturing advances towards automation and intelligence, the significance of OLAMP technology in composite placement is increasing. Motion programming for redundant AFP systems presents complexity and time-consuming challenges owing to kinematic redundancy, necessitating significant technical enhancements. Several new and effective OLAMP algorithms have emerged, aimed at cost reduction and facilitating more convenient operation for redundant motion planning. These algorithms effectively address numerous motion challenges encountered in the manufacturing process.



(a)



(b)

Fig. 2. (a) Design path (b)Layout path.

The traditional OLAMP methods solve the redundancy problem through the kinematic Jacobian's generalized inverse (pseudo-inverse) [19-21]. They use the null-space vector of the Jacobian matrix and employ the gradient-projection method to minimize specific kinematic metrics, such as manipulability [19], manipulator-velocity ratio (MVR) [20], torque [21], etc. However, this technique involves complex calculations related to pseudo-inverse, and it is challenging to generate solutions that simultaneously consider multiple constraints such as actuator velocity, acceleration, and collision avoidance.

There are other approaches to solving the redundancy problem by employing intelligent algorithms that optimize the relevant objectives. FarzanehKaloorazi et al. [22] took the Jacobian matrix determinant as a metric to quantify the singularity. They used particle swarm optimization (PSO) with Newton's method to obtain the singularity-free joint motion for a 13 DOFs coordinated robotic work cell. Similarly, Doan and Lin [23] used the PSO algorithm for redundancy optimization and robot placement by considering robot joint, singularity, and collision avoidance constraints simultaneously. Debout et al. [24] optimized the joint motion of the 7 DOFs redundant layup system by minimizing the joint curve curvature using the Levenberg-Marquardt method and a linear search procedure. Liao et al. [25] introduced a novel approach for assessing surface profile errors in robotic milling. They proposed the WOA* algorithm, which optimizes robot posture and workpiece orientation through selective weight calculation, ensuring superior convergence for enhanced efficiency and accuracy. Additionally, in their another study [26], a new stiffness metric considering robotic rotational deformations was proposed, and machining stiffness was optimized by employing clustering algorithms and greedy algorithms to solve for robot redundancy and workpiece positioning. Sridhar Reddy et al. [27] utilized the Multi-objective Particle Swarm Optimization (MOPSO) method to optimize the motion of a 9 DOFs welding robot, with the objective of minimizing positional and orientational errors to improve manipulator reachability. Liu et al. [28] developed a Levenberg-Marquardt surrogate model and a trimmed sequential linear programming method to solve the smooth motion paths of redundant robots. Peng et al. [29] introduced a global performance metric for robot path smoothness under joint acceleration constraints, utilizing a sequential linearization programming approach to derive motion paths for redundant milling robots that yield better surface quality and higher processing efficiency. Unlike traditional algorithms, optimization based on intelligent algorithms generally targets more generic problem descriptions, making it more friendly for structuring the problem. However, this approach also has some limitations. For different computational models, algorithm parameters need adjustment; otherwise, optimal solutions may not be efficiently attained. Moreover, as computational models become more complex, adjusting algorithm parameters becomes increasingly difficult, leading to low convergence accuracy or difficulties in convergence.

With the enhancement of computational power, there is an increasing prevalence of OLAMP methods based on Graph Search Algorithm for redundancy optimization. Dijkstra's shortest path technique has been widely used in robot joint motion planning [12-14]. The main steps of the algorithm based on Dijkstra's shortest path are [12]: First, the original continuous problem is converted into a discrete one, and inverse kinematic calculation generates a set of candidate joint configurations at each task point. Then, the shortest path on the corresponding graph searched by Dijkstra's algorithm is the desired trajectory. However, as pointed out in Refs. [30], the joint discrete step needs to be decreased for a smoother joint motion to be generated, which increases the graph nodes, leads to numerous computations, and is unaffordable and time-consuming for the shortest path search. Based on this, Gao et al. [31] applied a two-stage dynamic programming algorithm to improve the search efficiency of paths in the graph, including a 'global rough search' with the larger step in the initial stage and a ' local fine search' with the smaller step in the subsequent stage. The algorithm eliminates the joint nodes that violate the kinematic constraints and finds the shortest path representing the time-optimal and collision-free motion trajectory. Rakita et al. [14] also developed a method called Stampede, which also constructs the pose optimization problem as a graph-search problem in a discrete space, where the nodes in the graph were individually solved by nonlinear optimization, and a dynamic programming algorithm was adopted to find the optimum. Dai et al. [13] presented an effective trajectory planning method for robots with a small jerk. They first determined the initial trajectory by graph search and then adopted the greedy algorithm to optimize the trajectory by locally applying adaptive filters in the regions with large jerks. Malhan et al. [33] introduce an iterative graph construction method for determining trajectories for semi-constrained Cartesian paths. The algorithm leverages information from the Cartesian space to prioritize poses that yield higher-quality solutions. Additionally, a biasing scheme is devised to selectively sample the initial Cartesian poses from the available choices. Miteva et al. [34] constructed a weighted directed graph representing all possible ways of planning motion path. They then utilized a graph search algorithm to find the path with the minimum execution time among those constructed in the graph. The global optimization algorithm proposed by Lu et al. [32] combined Dijkstra's shortest path technique and DE algorithm, which speeds up the global smooth computation and avoids getting trapped in the local optimum under the collision-free constraint. The graph-based search algorithm tends to prioritize global optimization, reacting swiftly to environmental changes and employing a more straightforward path search approach, typically yielding superior paths. Nevertheless, these methods are susceptible to spatial discretization, resulting in poor path continuity. In complex environments, reducing the step size significantly diminishes search efficiency. Furthermore, as the dimensionality of joint space increases, the search performance of such algorithms deteriorates markedly.

In conclusion, many OLAMP techniques consider mechanical kinematic constraints, collision avoidance, singularity avoidance, etc. These techniques improve manufacturing efficiency and safety while avoiding complicated teach-in and online programming. Current redundancy optimization research predominantly revolves around establishing algorithms pertaining to three main directions: those based on Jacobian pseudo-inverse, intelligent algorithms, and graph search. However, as the environment and machining surfaces grow more intricate, or system redundancy increases, these methods frequently encounter challenges including complex modeling, limited generality, and inefficiency. Besides, as mentioned earlier, the Sample-based method can plan paths rapidly in complex environments. However, it cannot be directly used for redundancy optimization since it fails to consider that the Cartesian tasks are ordered and the task constrains the joint configuration. Moreover, the efficiency and path quality of the Sample-based method for finding paths in complex environments should be further improved.

Therefore, this paper improves the Sample-based motion planning algorithm for fully utilizing manipulator redundancy to obtain smooth collision-free and singularity-free joint motion path in complex layup environments.

1.2. Contributions & outline

This paper proposes a redundancy optimization algorithm aimed at resolving the challenge of smooth joint motion planning for 7 DOFs redundant manipulators during continuous layup tasks in complex environments. The algorithm improves the RRT* method to address redundancy optimization, incorporating several key improvements:

(1) Region Sampling Strategy: Task coordinates constrain joint sampling, so a region sampling strategy based on Gaussian distribution, termed PGSS, is introduced. PGSS allows simultaneous sampling in both joint space and task space.
(2) Guidance of Tree Growth: PGSS addresses the complexity of sampling in the search map with complex channels. It guides the

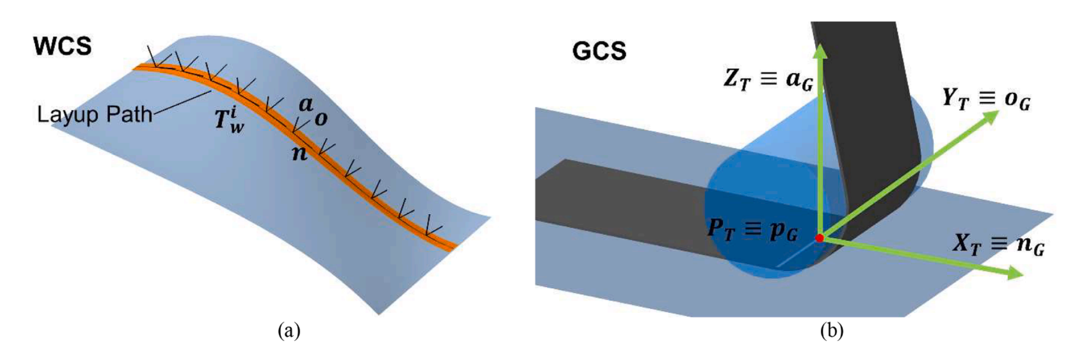


Fig. 3. (a) Discrete layup tasks. (b) Coordinate of machine head relative to the surface.

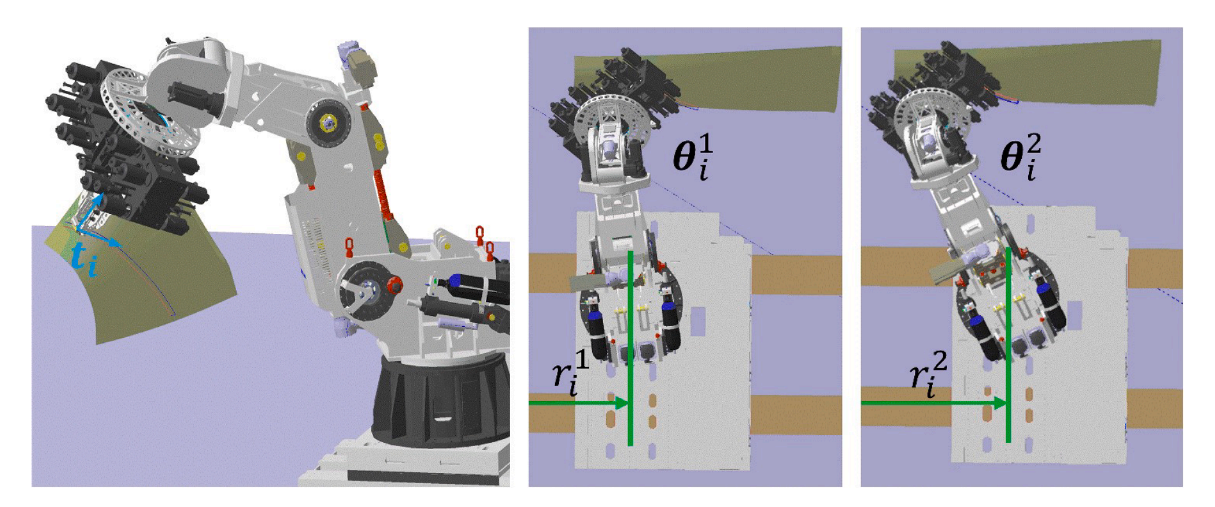


Fig. 4. Different redundancy schemes.

direction of tree growth by adjusting the sampling probability in different regions, thereby enhancing planning efficiency and optimizing the tree structure.

(3) Improved Steering and Local Optimization: Given that tasks must be executed sequentially, enhancements are made to the Steering and Local optimization strategies for joint motion planning constrained by intermediate tasks.

(4) Path Cost Formulation: The path cost function incorporates velocity, acceleration, and joint critical index considerations. This design aims to achieve smooth joint motions while preventing joint overtravel.

(5) Path Smoothing Technique: Polynomial fitting is employed as a path smoothing technique to reduce sharp corners in the joint path.

2. Problem formulation

2.1. Fiber placement task description

In fiber placement, the compaction roller is mounted at the fiber placement head and lays the tows on the mold surface. The fiber placement task can be described in the workpiece coordinate system (WCS) by a 4 × 4 matrix $T_W = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix}$, where p is the path point position and n, o, a are the path tangential, binormal and normal directions, respectively. In order to describe the layup path with arbitrary spatial morphology, CAM often discretizes the layup path into several task frames $T_W^i, i = 0, 1, \dots, N$, where $N + 1$ is the total number of path points [31] (as shown in Fig. 3a).

Let the transformation matrix between WCS and the global coordinate system (GCS) be M_G^W, then the fiber placement task is described in GCS as

$$T_G = M_G^W \cdot T_W \tag{1}$$

where $T_G = \begin{bmatrix} n_G & o_G & a_G & p_G \\ 0 & 0 & 0 & 1 \end{bmatrix}$. To ensure the compaction and layup accuracy, the tool coordinate system (TCS), which is fixed at the center of the roller and described by $T_{tool} = \begin{bmatrix} X_T & Y_T & Z_T & P_T \\ 0 & 0 & 0 & 1 \end{bmatrix}$, should be coincident with the task frame, as shown in Fig. 3b.

2.2. Kinematic redundancy description

Assuming the 7 DOFs serial manipulator joint coordinates as a 7-dimensional vector θ, the description of TCS under GCS can be obtained according to the forward kinematic (FK) modeling as:

$$T_{tool} = g(\theta) = M_G^{Base} M_{Base}^1(\theta_1) \left(\prod_{j=2}^{7} M_{j-1}^j(\theta_j) \right) M_7^{tool} \tag{2}$$

where $g(\cdot)$ represents the FK function of the system, M_G^{Base} is the transformation matrix of the manipulator base coordinate system (BCS) with respect to the GCS. Particular expressions for the matrix M_{Base}^1, M_{j-1}^j and M_7^{tool} can be obtained using the Denavit-Hartenberg (DH) technique [35], where they are expressed as a combination of the elementary rotations and translations $M_{j-1}^j(\theta_j) = R_X(\alpha_{j-1}) \cdot D_X(a_{j-1}) \cdot R_Z(\theta_j) \cdot D_Z(d_j)$ depending on the joint variable θ_j and the parameters a_{j-1}, d_j, α_{j-1} describing the link/joints geometry.

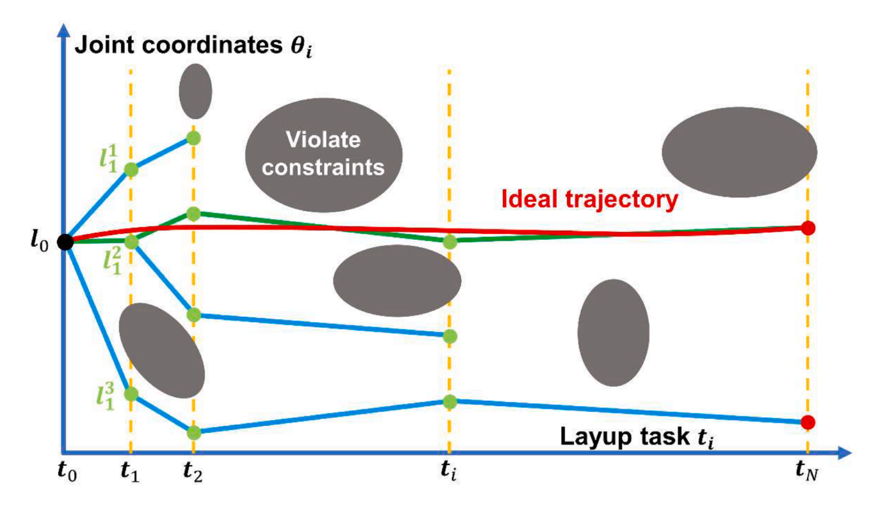


Fig. 5. The tree expands from the initial task (t_0) to the end task (t_N). (Gray indicates joint configurations that violate manipulator constraints).

Let θ_i describe the joint coordinates at a layup task, where $i = 0, 1, ..., N$ is the task number. As shown in Fig. 3b, the layup process requires that

$$g(\theta_i) = \mathrm{T}_G^i \tag{3}$$

T_G^i can be represented by the six-dimensional vector $\mathrm{t}_i = (x, y, z, \alpha, \beta, \gamma)$. Here, the placement orientation is described in Euler angle transformations (α, β, γ), and (x, y, z) is the placement position. Therefore, Eq. (3) shows that six-dimensional vectors correspond to seven joint variables. For universal series manipulators, if any one joint value r_i in θ_i is specified, the finite joint coordinates can be determined by the inverse kinematic (IK) calculation. For the unique mapping, IK has to take into account the configuration index c that corresponds to the manipulator posture, which specifies the shoulder, elbow and wrist configurations. Then, the joint coordinates θ_i corresponding to the given t_i can be obtained as follows:

$$\theta_i = g^{-1}(\mathrm{t}_i, r_i, c) \tag{4}$$

where $g^{-1}(\cdot)$ is the IK model. Thus, r_i, which is the redundant joint, is also a continuous variable. Let $\theta_i^k = g^{-1}(\mathrm{t}_i, r_i^k, c)$, $r_i^k \in \mathrm{R}_i$, where k denotes different redundancy schemes as shown in Fig. 4. R_i represents the feasible range of the redundant joint, and it is determined by multiple constraints which will be detailed in Section 3.1. Therefore, by varying r_i, different joint configurations corresponding to the same layup task can be obtained.

Let $\mathrm{l}_i = (\mathrm{t}_i, \theta_i)$, and $\mathrm{L} = \{\mathrm{l}_0, \mathrm{l}_1, ..., \mathrm{l}_N\}$ denote the sequence of the task frames and the corresponding joint coordinates. t_i is given by the discrete layup path and θ_i is solved by the proposed algorithm. Then, the problem can be described as follows: Given a sequence of layup path $\mathrm{T} = \{\mathrm{t}_0, \mathrm{t}_1, ..., \mathrm{t}_N\}$, the objective is to find a joint-smooth motion sequence $\mathrm{L} = \{\mathrm{l}_0, \mathrm{l}_1, ..., \mathrm{l}_N\}$ with no singularity and collision under the restriction on the redundant joint $r \in \mathrm{R}$.

2.3. Constraints

Redundant joints can be employed to deal with the practical constraints of the manipulator, which can be generally listed as follows.

- First, the trajectory planning for the 7 DOFs manipulator must handle the joint limitations, which can be written as,

$$\theta_j^{\min} \le \theta_j \le \theta_j^{\max}, j = \{1, ..., 7\} \tag{5}$$

Here, $\theta_j^{\min}$, $\theta_j^{\max}$ are the boundaries of the jth joint in the angle/displacement domain.

- The manipulator application faces the singularity problem. At the singularity, the manipulability index defined as $\sqrt{\det(J(\theta_i)J^T(\theta_i))}$ is equal to zero [19]. Therefore, to avoid singularities, the condition is

$$\sqrt{\det(J(\theta_i)J^T(\theta_i))} \ge \sigma_s \tag{6}$$

where θ_i is the joint coordinates at i^{th} task, $J(\theta_i)$ is the Jacobi matrix, and σ_s is the effective value for singularity avoidance.

- In motion planning, another significant challenge is collision avoidance. OBBs are commonly employed to represent objects, thanks to their ability to adapt to arbitrary orientations [36]. They play a vital role in collision calculations. Here, we utilize OBBs to approximate the shapes of manipulator joints and molds, aiding in collision detection. OBBs dynamically adjust with the motion of the joints, which is determined by the forward kinematic transformation associated with these joints. Subsequently, the Separating Axis Theorem (SAT) [36] is employed for collision detection between two OBBs. OBBs serve as efficient geometric approximations, facilitating rapid filtering before engaging in detailed collision detection. Following an OBB collision, the next step involves narrow detection of triangular patches within the collision region. This detection process operates on the Stereolithography (STL) representation of the detected objects.

2.4. Optimization objective

The proposed motion planning algorithm builds upon the RRT* algorithm framework. Originally, RRT* constructs a tree within the search space, subject to constraints on initial and final configurations, and iteratively optimizes this tree until an optimal solution is reached [16]. In our algorithm, adapted for AFP, paths connecting the initial and final laying tasks are predefined. These discrete laying tasks serve as constraints on the machine's motion during the laying process. While the joint sampling technique remains rooted in the original RRT*, the sampled joint coordinates must ensure the robot can fulfill the positional requirements of the given laying tasks. Thus, the search space construction encompasses two dimensions: layup tasks and joint configurations.

As shown in Fig. 5, the tree is expanded from the initial layup task (t_0) and then sampled in the joint domain determined by each task and manipulator constraints (as mentioned in Section 2.3). Some feasible solutions can be found when visiting the end task (t_N), and the cost-optimal solutions among them can be smoothed to obtain the ideal trajectory.

The cost is indispensable in the expansion of the tree. For the proposed algorithm, since the layup task are invariant, only considering the cost related to joints. Let $C_{tol}(\mathrm{l}_{i-1}, \mathrm{l}_i, \mathrm{l}_{i+1})$ denote the cost of three adja-

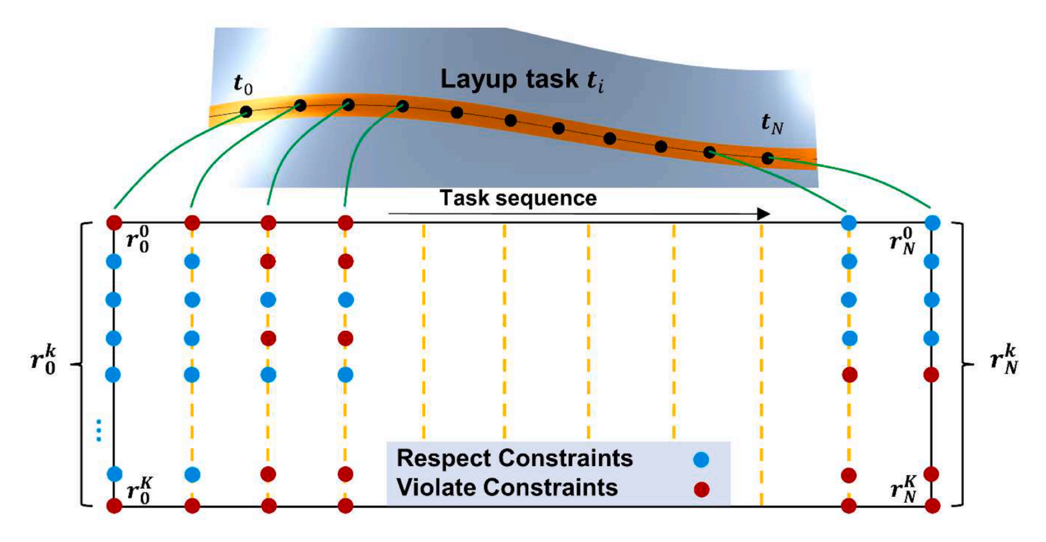


Fig. 6. Graph-based presentation of the discrete search map.

cent sampling points l_{i-1}, l_i, and l_{i+1}, where $l_i = (t_i, \theta_i)$, then

$$\begin{cases} C_{tol}(l_{i-1}, l_i, l_{i+1}) = \omega_V C_{Vlc}(l_{i-1}, l_i, l_{i+1}) + \omega_A C_{Acc}(l_{i-1}, l_i, l_{i+1}) + \omega_J C_{Joint}(l_{i-1}, l_i, l_{i+1}), & 0 < i < N \\ C_{tol}(l_0, l_1) = \omega_V C_{Vlc}(l_0, l_1) + \omega_J C_{Joint}(l_0, l_1), & i = 0 \end{cases} \tag{7}$$

where C_{Vlc} is the velocity cost, C_{Acc} is the acceleration cost, and C_{Joint} is the joint critical cost. ω_V, ω_A and ω_J are the weights between the three costs. The velocity cost describes the joint variation of the adjacent points and is expressed as

$$C_{Vlc}(l_{i-1}, l_i, l_{i+1}) = \text{sum}(\text{abs}(\theta_{i+1} - \theta_i) + \text{abs}(\theta_i - \theta_{i-1})) \tag{8}$$

where abs$(\cdot)$ denotes taking absolute values over vector elements and sum$(\cdot)$ denotes summing over vector elements. The acceleration cost describes the velocity variation of the joints as:

$$C_{Acc}(l_{i-1}, l_i, l_{i+1}) = \text{sum}(\text{abs}(\theta_{i+1} - 2\theta_i + \theta_{i-1})) \tag{9}$$

The joint critical cost evaluates the distance from the current joint to its limiting boundary, defined as

$$C_{joint}(l_i) = \frac{\| \theta_i - \theta_{mid} \|}{\| \theta_{mid} \|} \tag{10}$$

where $\theta_{mid} = \frac{\theta_{max} - \theta_{min}}{2}$, θ_{max} and θ_{min} are the vector composed of each joint's limits. $C_{Joint}(l_{i-1}, l_i, l_{i+1})$ in Eq. (7) represents the sum of the joint critical cost at the three adjacent points.

Defining the cost is always positive is essential for the planning algorithm. Thus, when the algorithm finds a feasible path, the total cost is

$$C_{tol}(l_0, l_N) = C_{tol}(l_0, l_1) + \sum_{i=1}^{N-1} C_{tol}(l_{i-1}, l_i, l_{i+1}) \tag{11}$$

where N is total number of tasks. Therefore, when the tree is expanded, the cost varies with the different sampling results of the joints. The velocity and acceleration reflect the motion smoothness, and the joint critical cost measures joint readjustment space. It is clear that minimizing the total cost function avoids layup vibrations, improves efficiency, and also ensures that all joints are away from their limits to accommodate positioning errors and further fine-tuning of the joints. Therefore, the optimization objective of the algorithm is

$$\text{Minmize}: \quad C_{tol}(l_0, l_N) \tag{12}$$

3. The redundancy optimization algorithm for 7 DOFs manipulator

This paper presents an RRT*-based redundancy optimization algorithm to solve the challenge of smooth motion planning for the 7 DOFs redundant manipulator in continuous tasks. The redundancy optimization process is structured into two main phases. First, a 'Map' delineating the feasible domain of the redundant joint for a given path is constructed based on manipulator constraints, as detailed in Section 2.3. Subsequently, the motion planning algorithm explores joint paths that aim to minimize the objective function, as outlined in Section 2.4, while adhering to the constraints imposed by the map and considering the sequence of layup tasks. The subsequent section will provide a comprehensive exposition of the proposed algorithm.

3.1. Map creation

The considered fiber placement system consists of 7 DOFs, and for a given 6-dimensional layup task t_i, there is exactly one redundant joint, whose value is denoted as r_i. Then, bringing r_i into the manipulator IK function (Eq. (4)) can calculate the unique joint coordinates θ_i for any given task under one configuration index $c = c_0$. Thus, the redundant joint can replace all-joints to construct the search space, which substantially reduces the search space dimensionality. This Two-dimensional space is called the search map.

Obviously, any joint variable contained in the vector θ_i can be considered as redundant, but it is often convenient to analyze by treating, for example, the robot base translational joint, the mold rotational joint, or the kinematic chain's first or end rotational joint as redundant. In addition, this paper constructs a map only calculating the inverse kinematics under a predefined one manipulator configuration index c_0, which realizes the unique mapping from task space to joint space. Also, the actual layup process will try to avoid the change of configuration index resulting in substantial joint variations.

To construct the map, r_i^k is sampled with step Δr within its joint limits

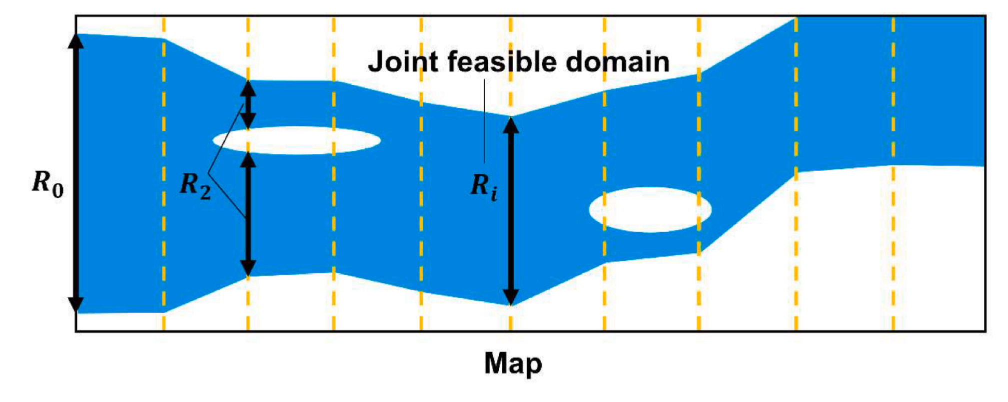


Fig. 7. The map discrebing the feasible domain of the redundant joint for the whole path.

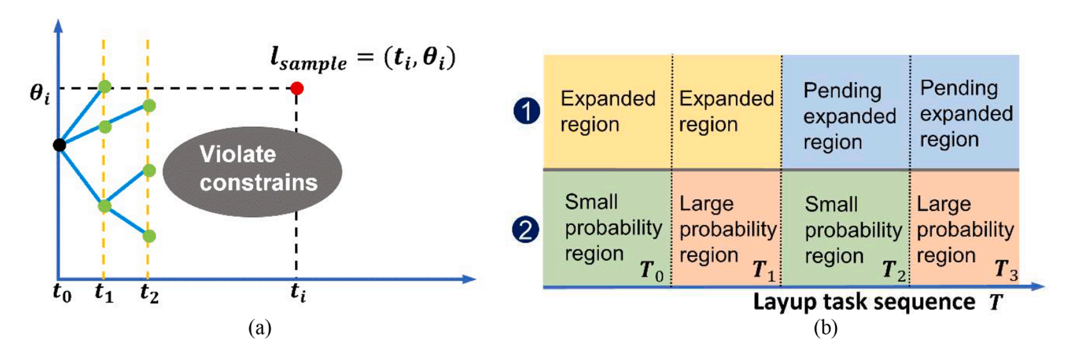


Fig. 8. (a) Sampling in the task space. (b) Division of the entire search space.

$[r_{\min}, r_{\max}]$, where $k = 0, 1, \ldots K$ denotes different redundancy schemes. Then, for a given layup task t_i, a set of possible joint states θ_i^k can be obtained by sequentially applying Eq. (4). These tasks are given in strict time order. The aforementioned process can be represented as shown in Fig. 6.

It should be noted that some configurations are excluded due to constraint violations. The remaining configurations are collision-free, singularity-free, and within the joint limits. Therefore, their corresponding r_i^ks constitute the feasible domain of the redundant joint at i^{th} task, and the domain is denoted as R_i. Then, the map is the one that represents the feasible domain distribution of the whole path (Fig. 7).

3.2. Motion planning

After generating the map, the optimization problem is to search within the map for the joint configurations sequence connecting the given tasks that minimize Eq. (12). This paper proposes an improved RRT* algorithm to solve this problem.

The motion planning process consists of three main parts:

(1) The feasible paths are found by rapidly expanding the tree using a piecewise Gaussian sampling strategy (PGSS).
(2) The tree is repeatedly expanded and optimized.
(3) The joint curve is smoothed with the polynomial fit.

The main framework of the proposed algorithm is shown in Algorithm 1.

Algorithm 1. The main framework.

1 **Initialization:** $\mathrm{T} = \{t_0, t_1, \ldots, t_N\}, V \leftarrow [l_0 = (t_0, \theta_0)], \mathrm{Map} = \{R_0, R_1, \ldots, R_N\}$
2 $h_{THR}, \varphi_{THR}, z_1, z_2, z_3$ //Parameters used for PGSS region delineation
3 **for** $ind = 1, \ldots, max_iter$
4 $l_{sample} \leftarrow \mathrm{PGSS}(\mathrm{T}, \mathrm{Map}, h_{THR}, \varphi_{THR}, z_1, z_2, z_3, ind)$ // Algorithm 2
5 $l_{nearest} \leftarrow \mathrm{Nearest}(V, l_{sample})$

(continued on next column)

(continued)

6 $l_{new} \leftarrow \mathrm{Steering}(l_{nearest}, l_{sample})$
7 **if** $\mathrm{WithinMap}(r_{new})$
8 $\mathrm{ConnectPrune}(V, l_{nearest}, l_{new}, B_{neighbor})$ // Algorithm 4
9 **if** $\mathrm{VisitGoal}(l_{new})$
10 $\mathrm{P} \leftarrow \mathrm{P} \cup \mathrm{Backtracking}(V, l_{new})$
11 **end**
12 **end**
13 **end**
14 $\mathrm{P}_{\min} \leftarrow \mathrm{argmin}(C_{tol}(\mathrm{P}))$

First, given task sequence $\mathrm{T} = \{t_0, t_1, \ldots, t_N\}$, where N is the total number of path points. Meanwhile, the parameter h_{THR}, φ_{THR} and z_1, z_2, z_3 related to the PGSS are given to regionalize the entire task sequence, the role of each parameter is described in detail in Section 3.2.1. Then, a feasible initial state needs to be first specified as the starting point, $l_0 = (t_0, \theta_0)$. The tree is initialized by adding the initial node l_0 to the tree. The map $\mathrm{Map} = \{R_0, R_1, \ldots, R_N\}$ representing the joint feasible domain is set. After that, the tree can be expanded in the search space to plan the optimal path (Algorithm 1, lines 3–13). The details of the algorithm are described below.

3.2.1. Sampling

In the expansion process, the random states are sampled first by PGSS (Algorithm 1 line 4), as shown in Fig. 8a. For the search space consisting of task space (horizontal coordinates) and joint space (vertical coordinates), PGSS samples in both task space and joint space, ensuring that the sampled joints are constrained by tasks.

The sampling process is shown in Algorithm 2. Firstly, $\mathrm{RegionDelineation}(\cdot)$ is performed on the task sequence T. This function divides the entire task into a task segment sequence $\mathrm{T_S} = \{T_0, T_1, \ldots, T_S\}$ according to the complexity of search map geometry, and assigns distribution parameters $\mu = [\mu_0, \mu_1, \ldots, \mu_S], \sigma = [\sigma_0, \sigma_1, \ldots, \sigma_S]$ and sampling frequency $\mathrm{F} = [F_0, F_1, \ldots, F_S]$ to each task segment, where each $T_s, s = 0, 1, \ldots S$ contains a series of continuous tasks (Algorithm 2 line 2). Then,

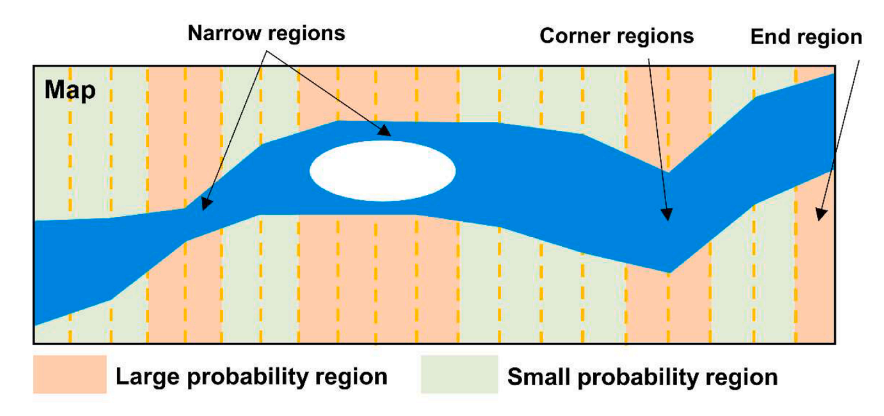


Fig. 9. Different sampling probability area division.

task sampling TaskSampling(·) samples once within each segment in segment order to obtain a layup location t_i (Algorithm 2 line 4). Accordingly, SampleRand(t_i, Map) performs joint sampling at the new position t_i. This function samples uniformly at t_i within the map to obtain redundant joint angles, and then invokes the IK calculation to obtain all joints θ_i (Algorithm 2 line 5). After that, a sampling state $l_{sample} \leftarrow (t_i, \theta_i)$ is generated (Algorithm 2 line 6).

Algorithm 2. $l_{sample} \leftarrow \mathrm{PGSS}(T, \mathrm{Map}, h_{THR}, \varphi_{THR}, z_1, z_2, z_3, ind)$.

```
1  if ind = = 1
2  (T_S, μ, σ, F, S)←RegionDelineation(T, Map, h_THR, φ_THR, z1, z2, z3)
3  end
4  t_i←TaskSampling(T_S, μ, σ, F, S) // Algorithm 3
5  θ_i←SampleRand(t_i, Map)
6  l_sample←(t_i, θ_i)
```

Next, the region sampling of PGSS is elaborated. This innovative approach involves partitioning the entire task sequence into distinct regions, each sampled with varying probabilities. By doing so, the sampled points can effectively converge towards the desired expanded points with heightened probability, consequently optimizing paths and enhancing the efficiency of path search operations.

3.2.1.1. Different probability regions of PGSS: divided according to the map height. As in Fig. 8b, the entire task can be divided in two ways: the first one is divided into the expanded region and pending expanded region according to the expansion of the tree, as in the upper part of Fig. 8b. Regions after sampling are called expanded regions, the new sampling points fall in the pending expanded region that will facilitate the tree to grow forward. Accordingly, this division is to facilitate the forward expansion of the tree so that a feasible path can be found rapidly.

The second one is to divide the whole task according to the geometric complexity of the search map into the large probability region and small probability region, as in the lower part of Fig. 8b. The points located in the large probability region will have a higher probability of being selected throughout the optimization process. In the motion planning, we expect more sampled points to be concentrated in the map's corner regions, narrow regions, and end region, as shown in Fig. 9. The corner and narrow regions characterize some undesirable joint motions that are prone to occur, such as corner regions corresponding to joint abrupt changes, and narrow regions with small feasible joint domains. Increasing the sampling rate in these places results in more points being sampled in these complex regions, which equates to providing more path options pass through these regions. By formulating a reasonable objective function, the probability of undesired motion occurring in these regions is reduced by allowing the algorithm to find the paths that have the best performance among them. The purpose of designating the end regions as high probability regions is to make more sampling points fall at the end tasks so that more complete paths connect from the start task to the end task, which is conducive to improving the quality of the final output paths.

The large probability region can be obtained by the information of the map height direction. The height of the map h_i located at t_i can be expressed by the range of the feasible redundant joint value, that is, $h_i = r_i^{\max} - r_i^{\min}$, where $r_i^{\max}$ and $r_i^{\min}$ are the maximum and minimum values of the continuous interval of the redundant joints. The larger value of h_i indicates that the position is reachable by the machine in more poses, and a smaller value indicates that the machine has fewer reachable poses for that position. If there are multiple consecutive intervals at this task, the total height is the sum of each segment's height. If the current height is less than the set threshold h_{THR}, then the region is defined as a narrow region, i.e.:

$$\left[\left(i|_{h_i<h_{THR}}\right) - z_1, \left(i|_{h_i<h_{THR}}\right) + z_1\right] \tag{13}$$

Among these, as the value of h_{THR} increases, more high probability regions are delineated, which can enhance the quality of path generation. However, it may somewhat impact the speed of calculation. $z_1 \in N_+$ is the set region transition width. It is used to expand a narrow region at the current task by a certain distance in both directions.

The paper provides the determination formula for parameter h_{THR} as follows:

$$h_{THR} = \left(1 - \frac{h_{\mathrm{mean}} - h_{\min}}{h_{\max} - h_{\min}}\right) \cdot h_{mean} \tag{14}$$

$h_{\max}$ represents the maximum height in the current search map, $h_{\min}$ represents the minimum value, and h_{mean} represents the average height of all path points contained within the current search map. The expression ($E = \left(1 - \frac{h_{\mathrm{mean}} - h_{\min}}{h_{\max} - h_{\min}}\right)$) in Eq. (14) reflects the proportion of narrow intervals to the total interval length. For example, when the mean height of the map h_{mean} approaches the minimum height $h_{\min}$, indicating a larger proportion of narrow regions, then E increases. Consequently, h_{THR} tends to approach the h_{mean}, suggesting an inclination towards expanding the threshold of narrow regions. Conversely, when the h_{mean} approaches the maximum height $h_{\max}$, signifying a smaller proportion of narrow regions, then E decreases. Consequently, h_{THR} significantly deviates from the h_{mean}, indicating a tendency to decrease the threshold of narrow regions.

Furthermore, the median deviation $\varphi_i = \left|r_i^{Med} - r_{i-1}^{Med}\right|$ is defined, where $r_i^{Med} = \left(r_i^{\max} + r_i^{\min}\right)/2$. r_i^{Med} is the median values of the feasible redundant joints for two adjacent laying tasks. The median deviation φ_i describes the continuity of the redundancy optimization space. Setting the deviation threshold φ_{THR}, Then the corner region is defined as:

$$\left[\left(i|_{\varphi_i>\varphi_{THR}}\right) - z_2, \left(i|_{\varphi_i>\varphi_{THR}}\right) + z_2\right] \tag{15}$$

So, the smaller the value of φ_{THR} is set, the more high-probability

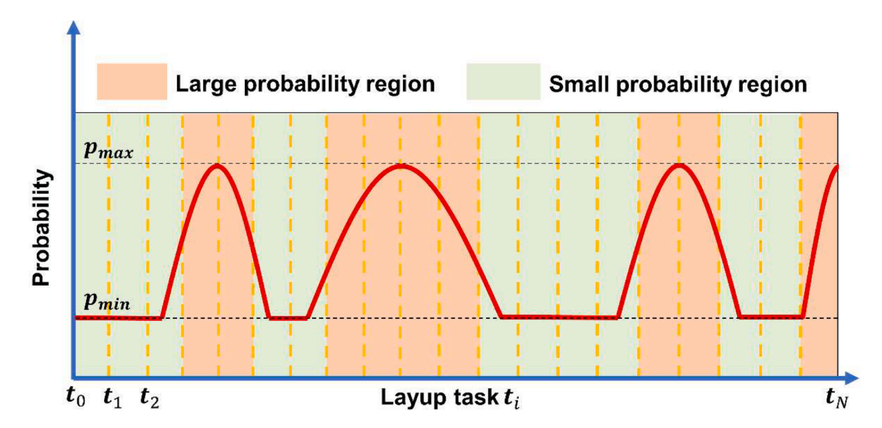


Fig. 10. Probability distribution.

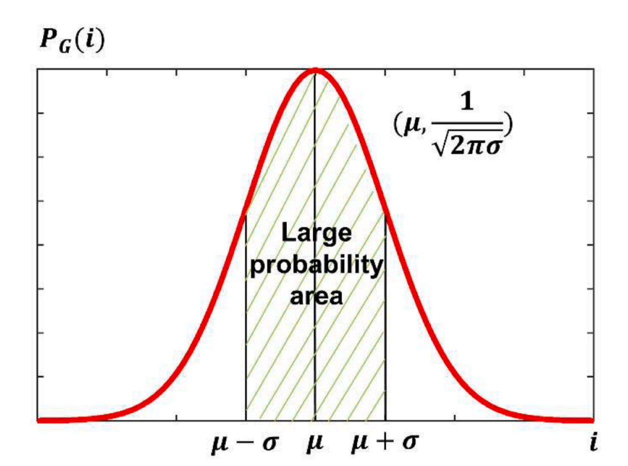


Fig. 11. Sampling probability distributions using the Gaussian sampling strategy.

regions there are, which is beneficial for improving the quality of the path. However, it may increase the calculation time. Actually, the value of φ_{THR} is more dependent on the motion performance of the AFP machine. If smaller joint fluctuations can easily cause laying defects, the threshold needs to be adjusted downward. $z_2 \in N_+$ is the set region transition width.

Therefore, considering the speed and acceleration constraints of redundant joints, we provide a method for determining φ_{THR}. Assuming the desired speed of the given machine tool is V_P, the maximum velocity constraint of redundant joints is ω_{MAX_r}, and the maximum acceleration constraint is a_{MAX_r}. If the redundant joint angles are all taken as median values r_i^{Med}, then the angular velocity is:

$$\omega_i = \frac{\varphi_i \cdot V_P}{\Delta P_i} \tag{16}$$

where ΔP_i is the distance between tasks t_i and t_{i+1} in Cartesian space. Then, the angular acceleration is

$$a_i = \frac{(\varphi_{i+1} - \varphi_i) \cdot 2V_P}{\Delta P_i + \Delta P_{i+1}} \tag{17}$$

We define the minimum value of φ_i where the angular velocity or acceleration of redundant joints exceeds the motion constraints under the set velocity V_P as the parameter values for φ_{THR}:

$$\varphi_{THR} = \{\varphi_i | (\omega_i > \omega_{MAX_r} \| a_i > a_{MAX_r}), i = 0, 1, ...N\} \tag{18}$$

Setting different values for V_P results in different values for φ_{THR}, depending on the desired working speed the robot is expected to achieve. The larger the desired speed, the smaller the value of φ_{THR}, and consequently, the greater the probability of high-density regions.

Finally, the end region is defined as: $[i - z_3, N]$, and t_N is the final task. It is worth mentioning that parameter z determines the width of the large probability region, but if the regions are adjacent or partially overlapping, they will be merged into one large probability region and the region width will be updated. As a result, the task sequence $T = \{t_0, t_1, ..., t_N\}$ is regionalized into large and small probability regions. The regions are numbered in time-order to obtain the task segment sequence $T_S = \{T_0, T_1, ..., T_S\}$ and its corresponding width of each segment Δ_s, where $s = 0, 1, ..., S$ is the task segment number.

3.2.1.2. Different distribution parameters of PGSS: determined by the region width. The proposed PGSS obeys a Gaussian distribution within the large probability regions and a Uniform distribution in the small probability regions. The whole probability distribution roughly as shown in Fig. 10, while $p_{\max}$ and $p_{\min}$ is the highest and lowest probabilities.

The parameter settings of PGSS are determined by the region width. It is known that Δ_s is the width of the task segment T_s. Let the distribution parameters μ_s, σ_s and the sampling frequency F_s be set for $T_s = \{t_i, t_{i+\Delta_s}\}$, where t_i is the starting task. The sampling probability function of Gaussian distribution is,

$$P_G(i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(i-\mu)^2}{2\sigma^2}\right) \tag{19}$$

Fig. 11 shows its probability distribution, where i is the task number and the sampling range $[\mu - \sigma, \mu + \sigma]$ is the large probability area. Then, if $T_s = \{t_i, t_{i+\Delta_s}\}$ is the large probability region, the distribution parameters will be set as $\mu_s = i + \frac{\Delta_s}{2}$, $\sigma_s = \frac{\Delta_s}{2}$. In this way, the sampled numbers will obey the Gaussian distribution $N(\mu_s, \sigma_s^2)$, and the large probability area of this distribution corresponds exactly to the T_s. The other side, if T_s is the small probability region that obeys the Uniform distribution, the distribution parameters $\mu_s = NULL$ and $\sigma_s = NULL$.

3.2.1.3. The sampling frequency of PGSS. In order to make the tree grow forward, PGSS will sample once in the region one by one in the region order. Since each region contains a different number of tasks, the peak probability of performing Gaussian sampling and uniform sampling is different for each region. Taking uniform sampling as an example, the fewer the number of tasks contained in the region, the higher the probability that the points in it are sampled. In order to make the overall sampling probability satisfy the curve shown in Fig. 10, different sampling frequencies $F = [F_0, F_1, ..., F_S]$ are set for each region. In order to clearly illustrate the setting of sampling frequency F_s, an example is given.

Suppose the layup task is divided into four segments $T_S = \{T_0, T_1, T_2, T_3\}$, where T_1 and T_3 are the large probability regions, T_0 and T_2 are the small probability regions. Then, the distribution parameters of T_1

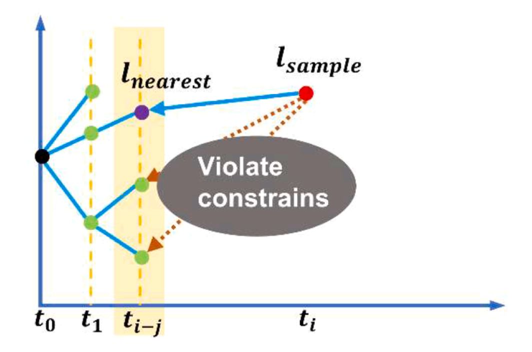


Fig. 12. The selection of the nearest point.

and T_3 are $\mu_1 = i_{T_1} + \frac{\Delta_1}{2}, \sigma_1 = \frac{\Delta_1}{2}$ and $\mu_3 = i_{T_3} + \frac{\Delta_3}{2}, \sigma_3 = \frac{\Delta_3}{2}$. From Eq. (19), the highest probabilities of the two Gaussian distributions are $p_1 = \frac{1}{\sqrt{2\pi\sigma_1}}$ and $p_3 = \frac{1}{\sqrt{2\pi\sigma_3}}$, respectively. Additionally,T_0 and T_2 obeys the Uniform distribution with the probabilities of $p_0 = \frac{1}{\Delta_0}$ and $p_2 = \frac{1}{\Delta_2}$, respectively. Then, the proportion f_s is calculated to modify the probability of each region to satisfy:

$$f_0 p_0 : f_1 p_1 : f_2 p_2 : f_3 p_3 = 1 : \varepsilon : 1 : \varepsilon \tag{20}$$

where $\varepsilon = \frac{p_{\max}}{p_{\min}}$ is the given ratio of the highest and lowest probabilities (Fig. 10). Therefore, to coordinate the sampling probability of different regions and stabilizes the probability ratio, the sampling frequency F of every region is obtained from the maximum iterations number max_*iter* and ratio $\mathrm{f} = [f_0, f_1, \dots, f_s, \dots, f_S]$.

$$\mathrm{F} = \mathrm{f} \cdot \max_iter \tag{21}$$

In summary, the task sampling strategy TaskSampling(·) employed by PGSS can be encapsulated as Algorithm 3: The Map is divided into S sub-regions by regionalization. In each iteration, the segment number s is modified to guide the tree from the extended region to the pending extended region (Algorithm 3 line 1). The small probability region performs uniform sampling (Algorithm 3 line 6) and the large probability region performs Gaussian sampling (Algorithm 3 line 8), while reducing the sample frequency (Algorithm 3 line 10). Note that when $F_s = 0$, then priority is forwarded to the next nearest segment for task sampling (Algorithm 3 line 13), and if no such segment exists, then the task is sampled from the previous nearest segment (Algorithm 3 line 15).

Algorithm 3. $\mathrm{t_i} \leftarrow \mathrm{TaskSampling}(\mathrm{T_S}, \mu, \sigma, \mathrm{F}, S)$.

1	$s = \bmod(ind, S)$
2	**while** $s \in [0, S]$
3	**if** $\mathrm{F}(s) > 0$
4	$\mu = \mu(s)\sigma = \sigma(s)$
5	**if** $(\mu, \sigma) = NULL$
6	$\mathrm{t_i} \leftarrow rand[\mathrm{T_S}(s)]$
7	**else**
8	$\mathrm{t_i} \leftarrow GaussSample(\mathrm{T_S}(s), \mu, \sigma)$
9	**end**
10	$\mathrm{F}(s) = \mathrm{F}(s) - 1$
11	**break**
12	**else**
13	$s = \underset{g}{Min}(find(g > s \ \&\& \ \mathrm{F}(g) > 0))$
14	**if** $s = NULL$
15	$s = \underset{g}{Max}(find(g < s \ \&\& \ \mathrm{F}(g) > 0))$
16	**end**
17	**end**
18	**end**

When the loop reaches the final segment, s is modified to the first segment so that the tree can be optimized again (Algorithm 3 line 1). Thus, the sampling segment gradually translates forward as the RRT* tree expands, accelerating the discovery of new paths. Therefore, the strategy of sampling probability regionalization improves the optimization capability and efficiency of the algorithm.

3.2.2. Nearest

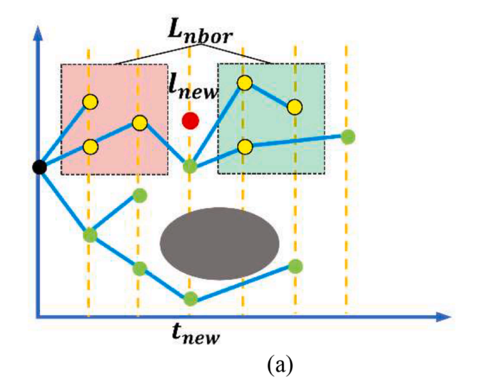
After sampling, a nearest state can be selected for the tree expansion (Algorithm 1 line 5), as shown in Fig. 12. l_{sample} denotes the current sampling point and (t_i, θ_i) is a description of the current task and joints. Starting from t_i and searching backward, find the closest extended lay task t_{i-j} to the $\mathrm{t_i}$ task. When t_{i-j} is determined, obtaining a scheme for the joint coordinates that already exist at t_{i-j}: $\theta^k_{i-j}, k = 1, \dots K_{i-j}$. Then, a nearest state $\mathrm{l}_{nearest}$ will be selected by comparing the joint variation of each scheme, which is represented as:

$$\mathrm{l}_{nearest} \leftarrow \left(\mathrm{t}_{i-j}, \underset{k=1,\dots,K_{i-j}}{\operatorname{argmin}} \left| \theta_i - \theta^k_{i-j} \right| \right) \tag{22}$$

where k represents the different redundancy schemes at t_{i-j}, K_{i-j} is the total number of schemes currently available. When a nearest state $\mathrm{l}_{nearest}$ is identified, then a new extension state can be determined by Steering, and the tree can be further expanded. Steering will be discussed in detail in Section 3.2.3.

3.2.3. Steering

After the nearest state is determined, the steering strategy can be executed and a new state is generated, denoted by $\mathrm{l}_{new} = (\mathrm{t}_{new}, \theta_{new})$ (Algorithm 1 line 6). The process of determining the tasks and joints within l_{new} is described below. The new task t_{new} is generated by extending the nearest point $\mathrm{t}_{nearest}$ one step forward, denoted by $\mathrm{t}_{new} = \mathrm{t}_{nearest+1}$. The calculation of the joint θ_{new} is related to $\mathrm{t}_{nearest}$ and t_{sample}. Assuming m denote the difference of the task number between $\mathrm{t}_{nearest}$ and t_{sample}, the redundant joint r_{new} in θ_{new} will consider the arc length d



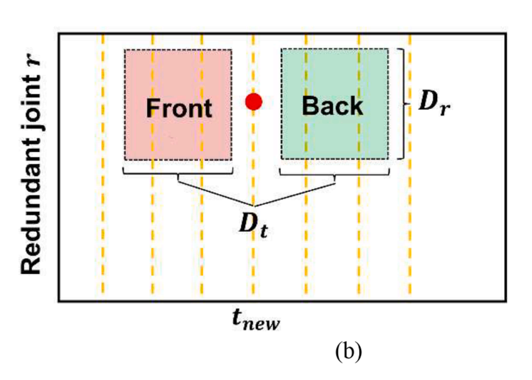
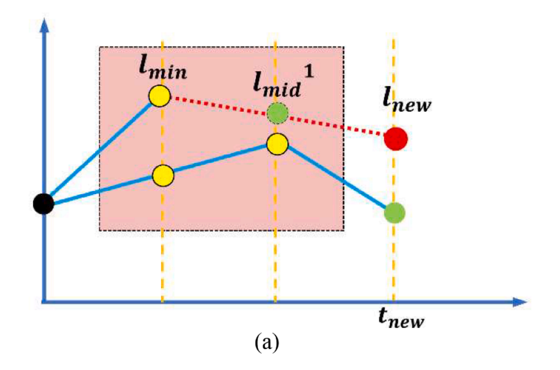


Fig. 13. Determine neighbor points. (a) Front and back neighbor points. (b) Boundary of the neighbor region.



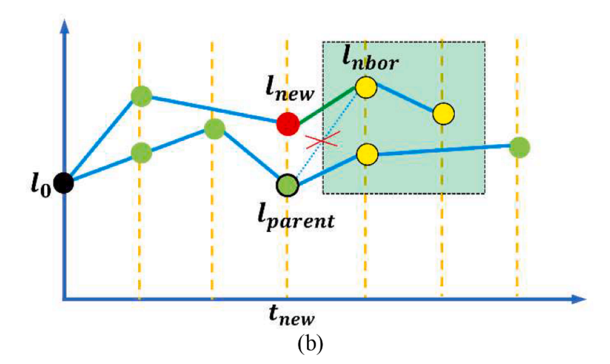


Fig. 14. Optimize the tree. (a) Connect the trees and generate intermediate points. (b) Prune the tree.

between $t_{nearest}$ and t_{sample}:

$$d_{nearest+j} = \sum_{j=1}^{m} \left| p_{nearest+j} - p_{nearest+j-1} \right| \tag{23}$$

where p is the position of the task.

The new angle r_{new} is calculated using a strategy where the angular displacement is proportional to the arc length. From Eq. (23), the arc length at $t_{nearest}$ is denoted as $d_{nearest}$. The arc length at t_{sample} is $d_{nearest+m}$, then r_{new} can be calculated as: $r_{nearest} + \frac{d_{nearest+1}}{d_{nearest+m}} \cdot (r_{nearest+m} - r_{nearest})$.The joint coordinates θ_{new} at new state can be obtained by IK calculation $g^{-1}(\cdot)$. Then, the new extended point can be expressed as:

$$l_{new} = \left[t_{new}, g^{-1}(t_{new}, r_{new}, c_o) \right] \tag{24}$$

When a new extension point is identified, the feasibility of the point will be detected by the map (Algorithm 1 line 7). The detection will be indexed to the redundant joint feasibility range R_{new}according to the task number, if $r_{new} \in R_{new}$, it means that the current state satisfies all manipulator constraints, then it will be inserted as a new point to connect or prune the node of the tree (Algorithm 1 line 8).

3.2.4. Optimize the tree

When a new point l_{new} is added to the tree, a new edge of the tree connecting the node l_{new} is created by performing a min-cost connection if the connection path satisfies the laying constraint. Meanwhile, the nodes close to l_{new} are optimized (Algorithm 4).

First, the neighbors on both sides of l_{new} will be determined, denoted by L_{nbor_front} and L_{nbor_back} (Algorithm 4 line 3). As shown in Fig. 13a, the red rectangle gives the forward region, and the yellow nodes in the region are the front neighbor points of l_{new}. Correspondingly, the nodes inside the green rectangle give the back neighbor points. The region boundary $B_{neighbor}$ is identified by specifying $D_t \times D_r$ area in the map (Fig. 13b), corresponding to the maximum difference between task number and the redundant joint, respectively.

Algorithm 4. ConnectPrune$(V, l_{nearest}, l_{new}, B_{neighbor})$.

```
1   l_min ← l_nearest
2   C_min ← C_tol(l_0, l_nearst) + C_tol(l_nearest, l_new)
3   (L_nbor_front, L_nbor_back) ← Neighbor(V, l_new, B_neighbor)
4   for ∀l_nbor ∈ L_nbor_front
5   if RefinedDetection(r_nbor, r_new) && C_tol(l_0, l_nbor) + C_tol(l_nbor, l_new) < C_min
6   l_min ← l_nbor
7   C_min ← C_tol(l_0, l_nbor) + C_tol(l_nbor, l_new)
8   end
9   End
10  V ← V ∪ (l_min, l_new)
11  for ∀l_nbor ∈ L_nbor_back
12  if RefinedDetection(r_new, r_nbor) && C_tol(l_0, l_new) + C_tol(l_new, l_nbor) < C_tol(l_0, l_nbor)
13  V ← (V / {(l_parent, l_nbor)}) ∪ (l_new, l_nbor)
14  C_tol(l_0, l_nbor) = C_tol(l_0, l_new) + C_tol(l_new, l_nbor)
15  end
16  End
```

Then, all possible connections from the front neighbors L_{nbor_front} to the node l_{new} will be considered, and the min-cost connection to l_{new} that satisfies the layup constraints therein will be selected as the best connection (Algorithm 4, lines 4–9). As shown in Fig. 14a, when a minimum cost node l_{min} is found, an edge from l_{min} to l_{new} will be generated. However, if l_{min} and l_{new} are not adjacent in task number, new intermediate points (e.g., $l_{mid}{}^{1}$) will be computed that satisfy the ordering of the layup tasks and are on the motion trajectory from l_{min} to l_{new}.

Let $l_{mid} = (t_{mid}, \theta_{mid})$ denote the intermediate point between l_{min} and l_{new}, where t_{mid} and θ_{mid} are descriptions of the task and joint of l_{mid}. To achieve the uniform transition of redundant joints, the redundant angle r_i of the intermediate points are related to the redundant angle r_{min} and r_{new}. The arc length is calculated by Eq. (23), and let L_{mid} denote the set of all intermediate points, then we have:

$$L_{mid} \leftarrow \left[t_{min+j}, g^{-1}\left(t_{min+j}, r_{min} + \frac{d_{min+j}}{d_{min+m}} \cdot (r_{min+m} - r_{min}), c_0 \right) \right], j = 1, \ldots, m \tag{25}$$

where m denotes the number of intermediate points between l_{min} and l_{new}.

To avoid collision between two neighboring task points, RefinedDetection$(\cdot)$ (Algorithm 4 lines 5) will be used during the connection process to check whether collision-free is satisfied in the motion between adjacent laying tasks. The specific method is as follows: (1) Take the joint coordinates of the adjacent tasks as the beginning and end points. (2) Use a small step size in the joint space to perform a linear discretization. (3) Judge whether the constraints are satisfied for all the discrete joints obtained. If the connections with all nodes in the neighborhood do not satisfy the constraints, the connections are disallowed and resampled.

Next, the new extension point l_{new} will be used to prune the tree (Algorithm 4 lines 11–15). The edges where the back neighbors L_{nbor_back} satisfy the condition will be modified (Algorithm 4 lines 12–14). For example, as shown in Fig. 14b, if the path from $l_0 \rightarrow l_{new} \rightarrow l_{nbor}$ is satisfying the constraints and the cost is less than the original cost from $l_0 \rightarrow l_{parent} \rightarrow l_{nbor}$, then a new edge will be generated from $l_{new} \rightarrow l_{nbor}$ and the old edge $l_{parent} \rightarrow l_{nbor}$ will be interrupted. If l_{new} and l_{nbor} are not adjacent, the new intermediate points will be generated with reference to Eq. (25).

After that, we need to determine if the current expanded node has reached the goal task, which is also the final task t_N. If a new path from the initial state to the goal is generated, the algorithm will backtrack the tree and find the path (Algorithm 1 lines 9–11). When the first path is found, the algorithm will continue to iterate and the tree keep being expanded and optimized until the algorithm terminates. Finally, a minimum cost path will be found by comparison (Algorithm 1 lines 3–14).

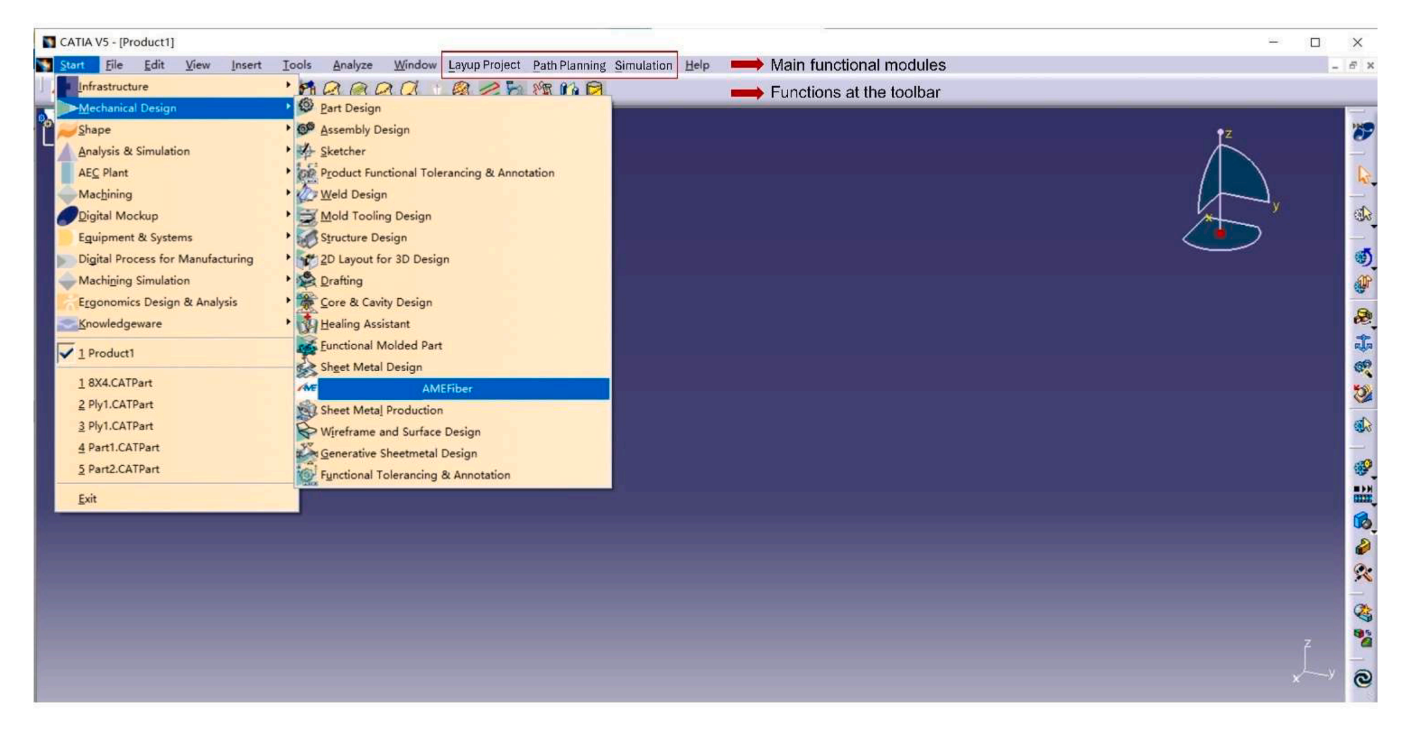


Fig. 15. Self-developed path planning software.

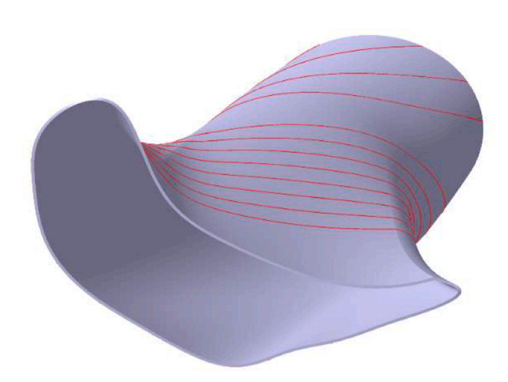


Fig. 16. The Aircraft air-inlet.

3.2.5. Smooth

Since the min-cost path obtained based on RRT* may contain corners and is not smooth enough, a polynomial fit is used to further smooth the generated path.

Minimizing the joint critical cost in the cost function reserves the adjustment space for the joints. For a given layup task, as all-joints vary with the redundant joint, the redundant joint r_i is considered as the optimization variable. Assuming that the redundant joint sequence of a path is denoted as $[r_0, r_1, ..., r_N]$, the path arc length is $[d_0, d_1, ..., d_N]$, then the fitted curve is:

$$r = \sum_{j=0}^{M-1} k_j d^j \tag{26}$$

where M is the polynomial order and the fitted values are $\hat{r}_i = \sum_{j=0}^{M} k_j d_i^j$. The unknown parameter k_j is solved using least-squares method such that:

$$k_j = \text{argmin}\left(\sum_{i=0}^{N} (r_i - \hat{r}_i)^2\right), j = 0, ...M - 1 \tag{27}$$

Since the fitted values must satisfy the map constraint $\hat{r}_i \in \text{Map}$, we



Fig. 17. 7 DOFs redundant fiber placement system.

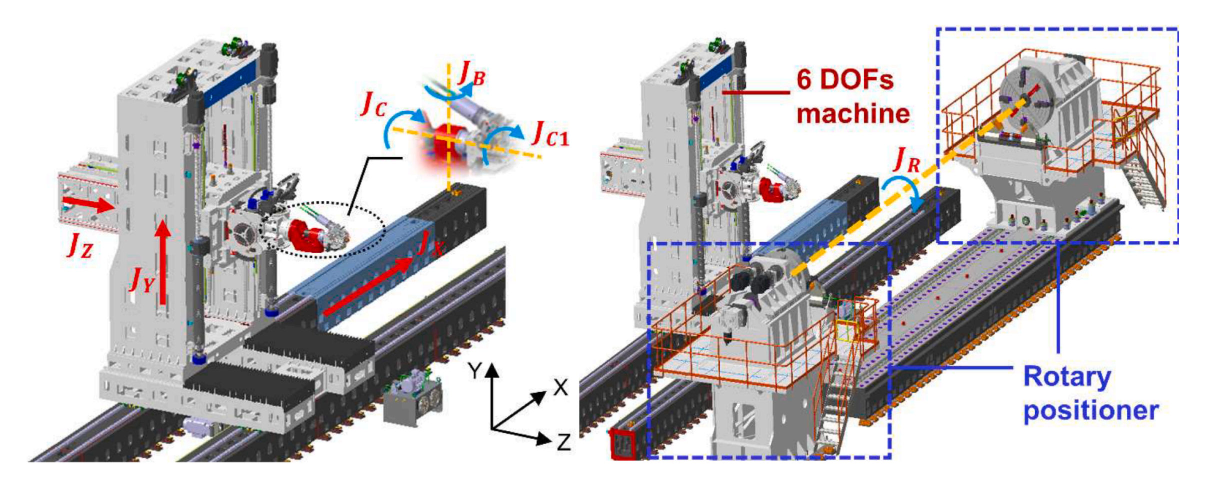


Fig. 18. Motion joints of the system.

Table 1
Joint constraint.

Joint	Joint Constraint
J_X	$[0mm, 12500mm]$
J_Y	$[0mm, 3000mm]$
J_Z	$[0mm, 7000mm]$
J_C	$[-180^\circ, 180^\circ]$
J_B	$[-90^\circ, 90^\circ]$
J_{C1}	$[-180^\circ, 180^\circ]$

Table 2
Comparison of the computation time.

	Classic algorithm (Dijkstra shortest path)	RRT*-based algorithm
$\Delta r = 3^\circ$	6.3 min (unsmooth)	4.1min
$\Delta r = 1^\circ$	40.4min	–
$\Delta r = 0.5^\circ$	114.6min	–

will make $M = 1, \ldots 8$ sequentially to change the curve shape until the constraint is satisfied. If it still cannot meet the requirement, the curve will be fitted in segments and transitioned with a first-order continuous boundary condition.

4. Experiment verification

To demonstrate the feasibility of the algorithm, the RRT*-based redundancy optimization algorithm proposed in this paper is applied to a 7 DOFs AFP system for laying up aircraft air-inlet.

4.1. An industrial use case: aircraft air-inlet

Aircraft air-inlet placement poses a significant challenge in the AFP domain. The air inlets are rotary structures, which are difficult to lay without the auxiliary mechanism. Hence, leveraging redundant equipment becomes crucial for successful air-inlet placement. In our experiment, we tested the layup of an aircraft air-inlet that can rotate alongside auxiliary rotary equipment. The curvature of the air-inlet surface undergoes dramatic changes, rendering the placement path highly complex and susceptible to collisions with the mold.

The fiber placement path was generated on the surface by the self-developed path planning software. The software utilizes the CATIA platform, based on CAA, and is developed using the C++ language. CAA provides a wealth of APIs for tasks such as generating curves on surfaces, creating equidistant curves, projecting curves, surface offsetting, etc. The developed software is named AMEFiber, which will be a standalone

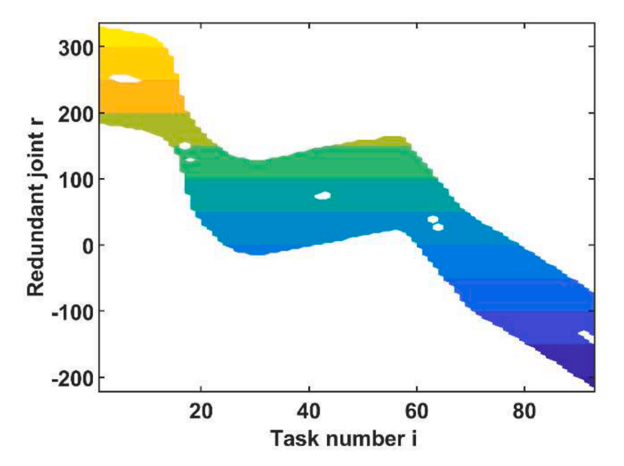


Fig. 19. Search map.

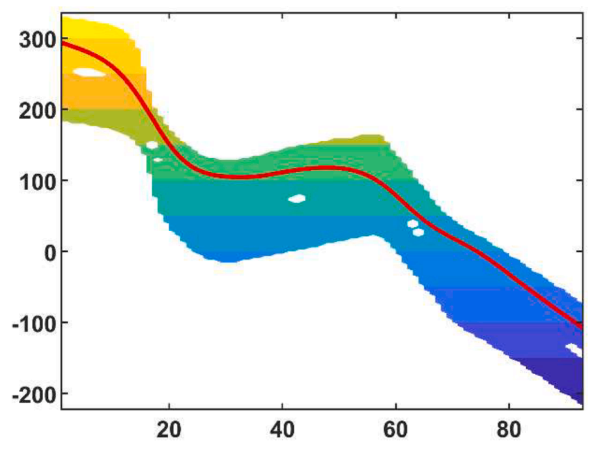


Fig. 20. The redundant joint profile after planning.

Workbench within CATIA for users to utilize. As shown in Fig. 15

The software comprises three main modules: project management, path planning, and post-simulation. In the path planning module, it first automatically creates a layup part based on the imported CPD or Fibersim file. Users can adjust parameters for the current layup, such as layup angle, gap, coverage rate, tow width, tow count, and path generation strategy, such as constant angle, geodesic line, and selected initial path points. The software then automatically generates center

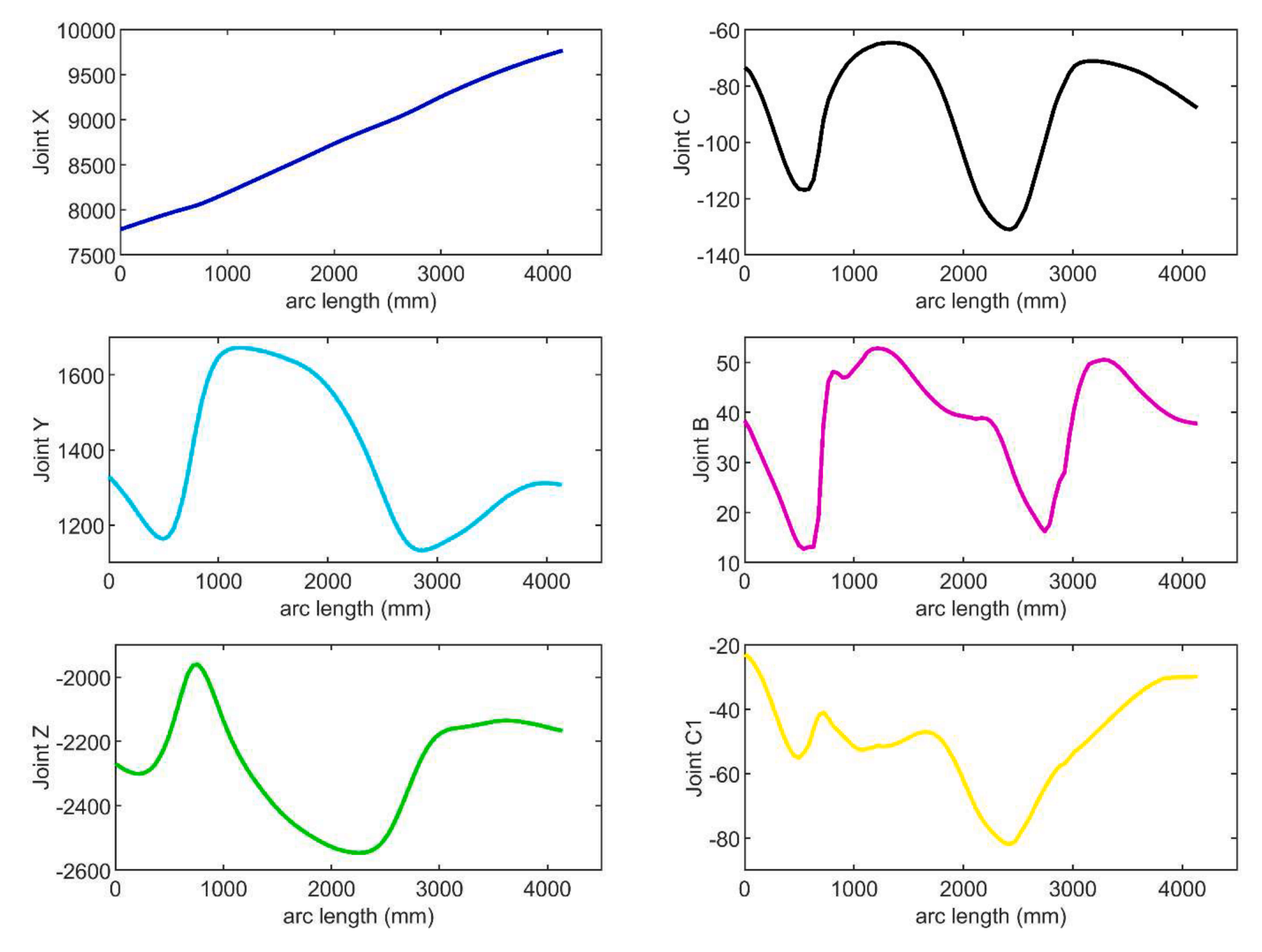


Fig. 21. Joint coordinate profiles.

paths (roller centers) and fiber paths on the surface according to the specified layup parameters. The discrete center paths, combined with cutting information, form the path file. The 45° center path generated on the inlet surface is illustrated in Fig. 16.

The post-simulation module takes the path file as input, constraints from the surrounding environment, and calculates the motion of each machine joint, thereby generating NC programs for machine motion. The motion planning algorithms in this paper are primarily integrated into this module. The simulation function takes NC programs as input, controls various joint values to visualize the machine layup motion, and provides collision interference, singularity verification, and other functions for user judgment.

Through the self-developed software, we can flexibly modify path generation strategies and post-processing strategies using integrated development algorithms, and utilize the CATIA platform to achieve visual and measurable optimization results of algorithms. The team has been developing this software for seven years, and its reliability has been effectively verified on the developed equipment.

After generating the central path and discretizing it during the path planning stage, the discretized layup tasks should be visited sequentially by the end-rollers in a smooth motion by utilizing the motion capabilities of the machine and the positioner in an optimal way, so the RRT*-based redundancy optimization algorithm will plan the joints path.

4.2. Implementation details

We applied the algorithm to a 7 DOFs redundant fiber placement system consisting of a 6 DOFs machine and a rotary positioner, as shown in Fig. 17. The motion joints of the system include three translational joints J_X, J_Y, J_Z and three rotational joints J_C, J_B, J_{C1} of the serial machine, and a rotational joint J_R around the *X*-direction of the positioner, as shown in Fig. 18. The joint constraints for the serial machine are listed in Table 1. Therefore, this system has 1 DOF redundancy with respect to the layup task.

The programming environment of the algorithm is Matlab in Win7SP1 using Intel® i5 @2.67 GHz 2.67 GHz. In order to compare the computation time with the different algorithm, the results obtained using Matlab are given in Table 2.

4.3. Process and analysis

The 45° fiber placement path are uniformly discretized into 93 points. The corresponding discrete layup task is denoted as t_i, where $i = 0, 1, ..., 92$.

To adopt the proposed motion planning algorithm, the positioner joint (J_R) coordinate is considered as the redundant variable *r*, which is discretized within its joint boundary. Then, the candidate discrete joint coordinates at each task point are obtained by applying the IK transformations accordingly. To ensure that no collisions, singularities and joint overtravels occur, each joint coordinate is verified. If it is detected that the current joint does not satisfy manipulator constraints, the corresponding redundant variable r_i^k is marked with an 'unacceptable' status. In this process, the original sequence of task points $T = \{t_0, t_1, ..., t_N\}$ is transformed into a search map consisting of tasks and feasible domains of the redundant joint (see Fig. 19). The obtained map is used for RRT*-based motion planning.

After the motion planning is completed, the final trajectory in joint space without collision and singularity is generated. From Fig. 20, the redundant joint satisfies the relevant constraints, and the joint coordinate profiles are quite smooth (see Fig. 21).

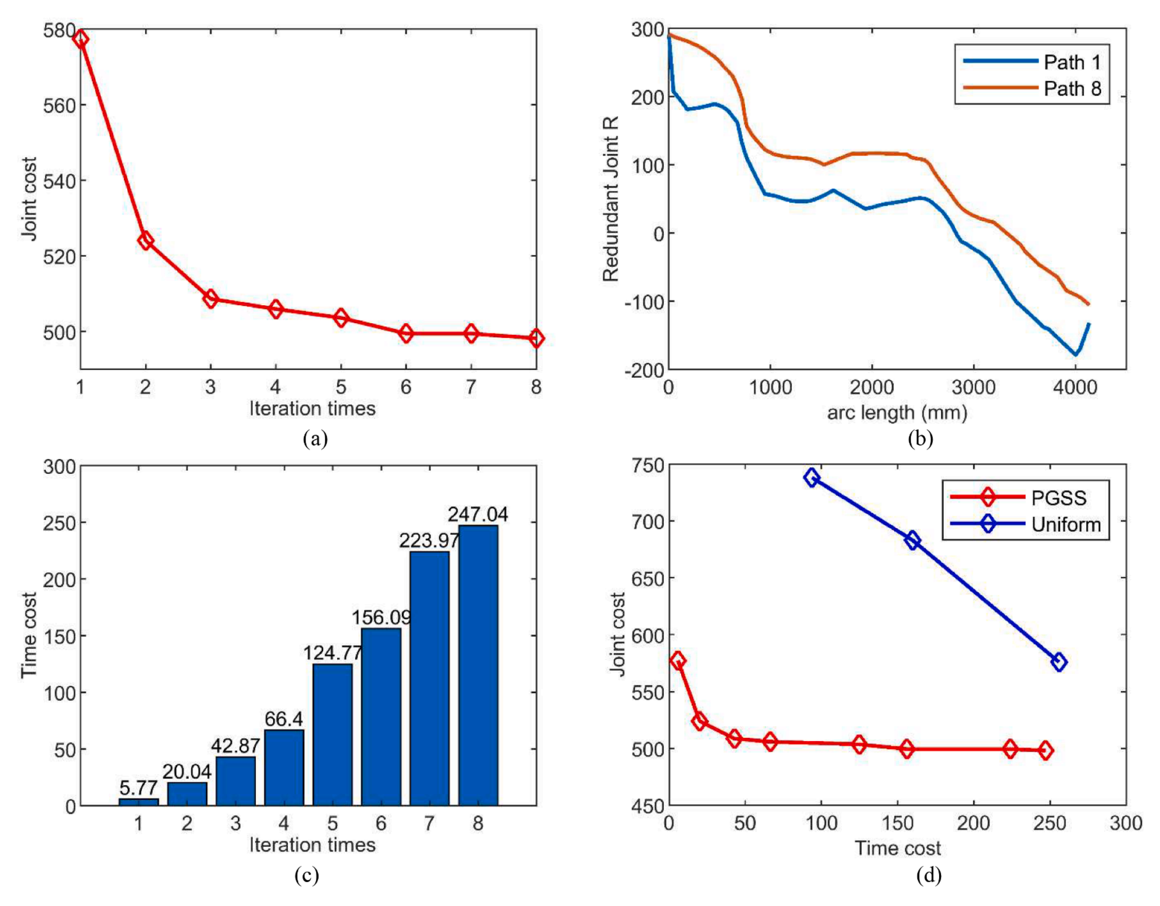


Fig. 22. (a) The joint cost of the motion planning. (b) Path comparison between the first planning and the eighth planning. (c) Time cost of each iteration. (d) Comparison between Uniform sampling and the proposed PGSS.

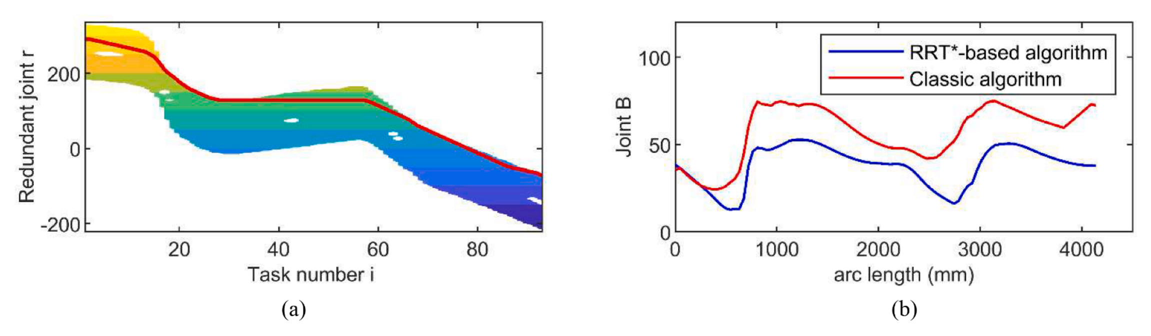


Fig. 23. (a) Redundant angle found by Dijkstra shortest path. (b) Comparison between classic algorithm and the proposed algorithm.

In Fig. 22a, the joint cost of the optimal trajectory is given, and it can be seen that the first planning joint cost is 577.2782. After the eighth planning, the cost is 498.1622. The cost has been reduced by 79.1160 from the first iteration to the eighth iteration. The results show that with the increase of the iteration, the joint cost of the path is gradually reduced. Fig. 22b shows the results of the redundant joint for the first planning and the eighth planning. It can be seen that Path 8 has better results than Path 1 because the joint is comparatively smoother, and it is beneficial to acquire a higher layup efficiency in AFP. Fig. 22c shows the results of time cost, and the cost of each iteration is given. From results, it could be seen that the time cost is 5.772 s when the first planning is completed. After that, as the path is gradually optimized, the time cost keeps increasing until the termination condition is satisfied.

Under the same conditions, the experimental results obtained using Uniform sampling strategy and PGSS sampling are shown in Fig. 22d. From the results, it can be found that when the path is searched for the first time using the Uniform sampling strategy, the time cost is 93.692 s. However, when using the proposed PGSS, the time cost is 5.772 s. The time cost has been reduced by 87.84 (93.7 %). In addition, the joint cost of the proposed PGSS is always lower than the Uniform sampling at the same time.

Finally, for comparison purposes, the considered problem is also solved using the Dijkstra shortest path algorithm. When setting the discrete step to $\Delta r = 3^{\circ}$, the redundant angle found by the algorithm is very close to the limit (Fig. 23a). Furthermore, influenced by joint discretization, the disparities in redundant joints are multiples of the step size, thereby affecting the optimization results of the algorithm. The larger the step size, the less smooth the results may be. Taking joint J_B as an example, the red line shows it tends to lead to sharp corners in the joint curve (Fig. 23b). In general, decreasing the discrete step Δr will improve this situation, but the computation time will increase significantly.

More details about the computational efficiency of the known and proposed algorithms are given in Table 2, which confirms the advantages of the developed technique to deal with such the complex surface.

5. Conclusion

Generating a smooth motion path for a redundant AFP system is crucial, especially in complex layup environments. This paper proposes an RRT*-based redundant optimization algorithm for planning continuous layup joint motions without collision and singularity for 7 DOFs manipulators.

To generate paths that satisfy layup constraints, we construct a search map describing the feasible domain of the redundant joint. This map delineates the complexity of the layup environment and is formed by eliminating collision and singularity joint configurations. The path's cost comprises velocity, acceleration, and joint critical index for joint smoothing. Several strategies are proposed to enhance RRT* for redundancy optimization. These primarily include: (1) Utilizing PGSS to sample in both joint and task space to ensure that the sampled joints adhere to task constraints. (2) Enhancing algorithm convergence rate and path quality in complex channels by dividing the search map into large and small probability regions based on its features. (3) Implementing Steering and Local Optimization. For non-adjacent points, the redundant joints of points between them are automatically planned for uniform motion, ensuring the continuity of the task sequence. (4) Finally, a path smoothing technique based on polynomial fitting is applied to reduce path sharp corners

The proposed algorithm follows an incremental expansion process, exhibiting characteristics of asymptotic optimization. To validate its efficacy, a 7 DOFs redundant machine is tested within the complex layup environment of an aircraft air-inlet. The planning results demonstrate the machine's ability to traverse paths while avoiding collisions and singularities, showcasing relatively smooth joint trajectories. Additionally, a comparison with the classical Dijkstra shortest path reveals that the proposed algorithm yields smoother joint profiles with higher efficiency.

Through experimental validation, the proposed algorithm demonstrates the following characteristics:

(1) Efficiently finding joint trajectories that adhere to layup constraints in complex environments, given the existence of feasible trajectories.
(2) Gradual smoothing of joint trajectories with increasing iterations.
(3) Enhanced optimization effectiveness and efficiency in complex layup environments.

Although experiments were conducted using a horizontal laying machine as an example, the proposed method is applicable to various continuous machining scenarios. These scenarios include instances where the equipment possesses one more DOF relative to the machining task, where the machining task is discrete and specified, where the machining paths are intricate, and where there are stringent requirements for motion continuity, such as in welding, milling or polishing applications.

CRediT authorship contribution statement

Qian Yang: Writing – original draft, Methodology, Conceptualization. **Weiwei Qu:** Writing – review & editing. **Yanzhe Wang:** Visualization, Software, Data curation. **Xiaowen Song:** Project administration. **Yingjie Guo:** Funding acquisition. **Yinglin Ke:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Funding acknowledgement

Funding: This work is supported by the National Natural Science Foundation of China [No.52105535].

References

[1] G. Dorey, Carbon fibres and their applications, J. Phys. D. Appl. Phys. 20 (1987) 245–256.
[2] D.H.J.A. Lukaszewicz, C. Ward, K.D. Potter, The engineering aspects of automated prepreg layup: history, present and future, Compos. Part B Eng. 43 (2012) 997–1009.
[3] A. Crocky, C. Grant, D. Kelly, X. Legrand, G. Pearce, Fiber placement processes for composites manufacture. Adv. Compos. Manuf. Process Des., 2015, pp. 79–92.
[4] G. Marsh, Automating aerospace composites production with fibre placement, Reinf. Plast. 55 (2011) 32–37.
[5] V. Agarwal, Trajectory planning of redundant manipulator using fuzzy clustering method, Int. J. Adv. Manuf. Technol. 61 (2012) 27–744.
[6] S. Zargarbashi, W. Khan, J. Angeles, Posture optimization in robot-assisted machining operations, Mech. Mach. Theory. 51 (2012) 74–86.
[7] M. Kang, H. Shin, D. Kim, S.E. Yoon, TORM: collision-free trajectory optimization of redundant manipulator given an end-effector path, arXiv preprint, arXiv: 1909.12517, (2019).
[8] L. Huo, L. Baron, The self-adaptation of weights for joint-limits and singularity avoidances of functionally redundant robotic-task, Rob. Comput. Integr. Manuf. 27 (2011) 367–376.
[9] J. Klosowski, M. Held, J.B.M. Mitchell, H. Sowizral, K. Zikan, Efficient collision detection using bounding volume hierarchies of k-DOPs, IEEE Trans. Vis. Comput. Graph. 4 (1998) 21–36.
[10] C. Tu, L. Yu, Research on collision detection algorithm based on AABB-OBB bounding volume, 2009 First Int. Workshop Educ. Technol. Comput. Sci. 1 (2009) 331–333.
[11] C. Faria, F. Ferreira, F. Ferreira, W. Erlhagen, S. Monteiro, E. Bicho, Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance, Mech. Mach. Theory. 121 (2018) 317–334.
[12] G. Erd˝os, A. Kov´acs, J. V´ancza, Optimized joint motion planning for redundant industrial robots, CIRP Ann. 65 (2016) 451–454.
[13] C.K. Dai, S. Lefebvre, K.M. Yu, J.M.P. Geraedts, C.C.L. Wang, Planning jerk-optimized trajectory with discrete-time constraints for redundant robots, IEEE Trans. Autom. Sci. Eng. (2020) 1–14.
[14] D. Rakita, B. Mutlu, M. Gleicher, STAMPEDE: a discrete-optimization method for solving pathwise-inverse kinematics, IEEE Int. Conf. Robot. Autom., IEEE (2019).
[15] S.M. La Valle, J.J. Kuffner, Randomized kinodynamic planning, Int. J. Rob. Res. 20 (2001) 378–400.
[16] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Rob. Res. 30 (2011) 846–894.
[17] O. Arslan, P. Tsiotras, Use of relaxation methods in sampling-based algorithms for optimal motion planning, IEEE Int. Conf. Robot. Autom., IEEE (2013).
[18] L. Janson, E. Schmerling, A. Clark, M. Pavone, Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions, Int. J. Rob. Res. 34 (2015) 883–921.
[19] T. Yoshikawa, Manipulability and redundancy control of robotic mechanisms, IEEE Int. Conf. Robot. Autom., IEEE (1985).
[20] R. Dubey, J. Luh, Redundant robot control using task based performance measures, J. Robot Syst. 5 (1988) 409–432.
[21] S. Nokleby, R. Fisher, R. Podhorodeski, F. Firmani, Force capabilities of redundantly-actuated parallel manipulators, Mech. Mach. Theory. 40 (2005) 578–599.
[22] M. FarzanehKaloorazi, I. Bonev, L. Birglen, Simultaneous path placement and trajectory planning optimization for a redundant coordinated robotic workcell, Mech. Mach. Theory. 130 (2018) 346–362.
[23] N. Doan, W. Lin, Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6R articulated robots, Rob. Comput. Integr. Manuf. 48 (2017) 233–242.
[24] P. Debout, H. Chanal, E. Duc, Tool path smoothing of a redundant machine: application to automated fiber placement, Comput. Aided Des. 43 (2011) 122–132.
[25] Z.Y. Liao, Q.H. Wang, Z.H. Xu, H.M. Wu, B. Li, X.F. Zhou, Uncertainty-aware error modeling and hierarchical redundancy optimization for robotic surface machining, Rob. Comput. Integr. Manuf. 87 (2024) 102713.
[26] Z.Y. Liao, Q.H. Wang, H. Xie, J. Li, P. Hua, Optimization of robot posture and workpiece setup in robotic milling with stiffness threshold, IEEE-ASME T. Mech. 27 (2021) 582–593.
[27] A.S. Reddy, V.S. Chembuly, V.K. Rao, Multi-objective optimization approach for Inverse Kinematics of redundant manipulator for welding applications, Mater. Today: Proceedings. (2023).

[28] Y. Liu, X. Chen, L. Wu, G. Huang, J. Luo, Y. Huang, P. Li, Global optimization of functional redundancy in a 6R robot for smoothing five-axis milling operations, Eng. Optim. 56 (2024) 138–154.
[29] J. Peng, Y. Ding, G. Zhang, H. Ding, Smoothness-oriented path optimization for robotic milling processes, Sci. China-Technol. Sc. 63 (2020) 1751–1763.
[30] Y.A. Lu, C.Y. Wang, J.B. Sui, L.J. Zheng, Smoothing rotary axes movements for ball-end milling based on the gradient-based DE method, J. Manuf. Sci. Eng. 140 (2018) 121008.
[31] J. Gao, A. Pashkevich, S. Caro, Optimization of the robot and positioner motion in a redundant fiber placement workcell, Mech. Mach. Theory. 114 (2018) 170–189.
[32] Y. Lu, K. Tang, C. Wang, Collision-free and smooth joint motion planning for six-axis industrial robots by redundancy optimization, Rob. Comput. Integr. Manuf. 68 (2021) 102091.
[33] R.K. Malhan, S. Thakar, A.M. Kabir, P. Rajendran, P.M. Bhatt, S.K. Gupta, Generation of configuration space trajectories over semi-constrained cartesian paths for robotic manipulators, IEEE T. Autom. Sci. Eng. 20 (2022) 193–205.
[34] L. Miteva, I. Chavdarov, K. Yovchev, Trajectory planning for redundant robotic manipulators with constrained joint space, IEEE 2020 Int. Conf. Softw., Telecommun. Comput. Netw., IEEE (2020).
[35] J. Denavit, A kinematic notation for lower-pair mechanisms based on matrices, Trans. ASME. J. Appl. Mech. 22 (1955) 215–221.
[36] J.W. Chang, W. Wang, M.S. Kim, Efficient collision detection using a dual OBB-sphere bounding volume hierarchy, Comput. Aided Des. 42 (2010) 50–57.